Package ‘statcomp’

August 25, 2016

Title Statistical Complexity and Information measures for time series analysis

Version 0.0.0.9000

Author Sebastian Sippel [aut, cre], Holger Lange [aut], Fabian Gans [aut]

Maintainer Sebastian Sippel <ssippel@bgc-jena.mpg.de>

Description An implementation of local and global statistical complexity measures for time series analysis in R. The package provides functions to compute statistical complexity and information measures for any given time series based on ordinal statistics (Bandt and Pompe 2002). The ordinal pattern statistics are used to calculate a variety of global (Permutation Entropy, MPR complexity) and local (Fisher Information) complexity and information measures (for further information, see Martin, Plastino and Rosso 2007; Olivares et al 2012). In addition, methods to derive weighted ordinal pattern distributions are supplied, where a user-specified weights-generating function can be selected (e.g. variance-weighted, Fadlallah et al 2013). Complexity and information measures constitute a simple, quick and powerful tool to classify and cluster (a large number of) time series, including for model-data comparisons.

Depends R (>= 2.7.0)

License GPL-2

LazyData true

Imports zoo

RoxygenNote 5.0.1

NeedsCompilation yes

R topics documented:

`adjust_pattern` *A function to create new pattern-coding schemes for the Fisher Information.*

Description

Adjusts and reorders a pattern ordering matrix.

Usage

```
adjust_pattern(target_pattern, ndemb)
```

Arguments

`target_pattern`	A numeric matrix that specifies the pattern to be transformed into the position vector. ATTENTION: Pattern should be in the ranks permutation notation, otherwise does not really make sense.
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb

Details

This function reorders permutations based on "jumps" or based on "bitflips".

Value

A numeric matrix that contains the permutation matrix.

Author(s)

Sebastian Sippel

References

Sebastian Sippel (2014). Master Thesis. University of Bayreuth.

fis	*A (low-level) function to compute the Fisher-information*

Description

The function computes the Fisher information, i.e. a local information measure based on two different discretizations.

Usage

```
fis(odp, discretization)
```

Arguments

opd	A numeric vector that details an ordinal pattern distribution in a user-specified permutation coding scheme.

Details

The Fisher information is a local information and complexity measure, computed based on the ordinal pattern distribution. The Fisher information is based on local gradients, hence it is sensitive to the permutation coding scheme. Options for discretization: "Olivares.2012" or "Ferri.2009", following Fisher Information discretization schemes in the respective publications.

Value

The normalized Fisher information measure in the range [0; 1].

Author(s)

Sebastian Sippel

References

Olivares et al (2012): Physica A 391 (2012) 2518–2526; Olivares et al (2012): Physics Letters A 376 (2012) 1577-1583; Ferri et al (2009): Phys. Lett. A 373 (2009) 2210–2214.

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
opd = ordinal_pattern_distribution(x = x, ndemb = 6)
fis(opd = opd)
```

`generate_lehmerperm_matrix` *A function to generate the Lehmer permutation ordering.*

Description

Generates all permutations of a given embedding dimension, ordered according to the Lehmer coding scheme.

Usage

```
generate_lehmerperm_matrix(ndemb)
```

Arguments

`ndemb`	The embedding dimension.

Details

This function converts ranks to indices and back.

Value

A numeric matrix that contains the Lehmer permutation pattern.

Author(s)

Sebastian Sippel

References

http://www.keithschwarz.com/interesting/code/?dir=factoradic-permutation

`global_complexity` *A function to compute global information and complexity measures for time series*

Description

This is a high-level function that calculates global complexity measures directly from a given time series or ordinal pattern distribution.

Usage

```
global_complexity(x = NA, opd = NA, ndemb)
```

Arguments

`x`	(OPTIONAL) If opd is not specified, a time series vector x must be specified
`opd`	A numeric vector that details an ordinal pattern distribution in a user-specified permutation coding scheme.
`ndemb`	(OPTIONAL) If x is given, the embedding dimension (ndemb) is required.

Details

This function calculates the following global measures of complexity and information:

- Permutation Entropy (PE, cf. Bandt and Pompe, 2002)
- Permutation Statistical complexity (MPR complexity, cf. Martin, Plastino and Rosso, 2006)
- Number of "forbiden patterns" (cf. Amigo 2010)

Value

A named vector containing the three global complexity measures.

Author(s)

Sebastian Sippel

References

Bandt and Pompe (2002): Physical Review Letters 88 (2002), 174102-1-174102-4. Martin, Plastino and Rosse (2006): Physica A 369 (2006) 439–462 Amigo (2010): Permutation Complexity in Dynamical Systems. Springer. ISBN 978-3-642-04083-2

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
global_complexity(x = x, ndemb = 6)
# or:
opd = ordinal_pattern_distribution(x = x, ndemb = 6)
global_complexity(opd = opd, ndemb = 6)
```

`hellinger_distance` *Distance measure between ordinal pattern distributions: Hellinger distance*

Description

Compute the Hellinger Distance

Usage

```
hellinger_distance(p, q)
```

Arguments

`p`	An ordinal pattern distribution
`q`	A second ordinal pattern distribution to compare against p.

Details

This function returns a distance measure.

Value

A vector of length 1.

Author(s)

Sebastian Sippel

References

none

Examples

```
hellinger_distance(p=ordinal_pattern_distribution(rnorm(10000), ndemb = 5), q= ordinal_pattern_distributio
```

`limit_curves` *Limit curves in the Entropy-Complexity plane*

Description

Compute the limit curves in the Entropy Complexity plane

Usage

```
limit_curves(ndemb, fun = "min")
```

Arguments

`ndemb`	Embedding dimension
`fun`	Whether the upper (max) or lower (min) limit curve should be computed

Details

This function returns the respective limit curve.

Value

A list with two entries

Author(s)

Sebastian Sippel

References

none

`MPR_complexity` *A function to compute the MPR-complexity*

Description

The function computes the MPR complexity, i.e. a generalized (global) complexity measure based on the Jenson-Shannon divergence.

Usage

```
permutation_entropy(odp)
```

Arguments

`opd`	A numeric vector that details an ordinal pattern distribution.

Details

Generalized complexity measures combine an information measure (i.e. entropy) with the distance of the distribution from the uniform distribution ('disequilibrium'). As a global measure, MPR-complexity is insensitive to the permutation coding scheme.

Value

The normalized MPR complexity measure in the range [0; 1].

Author(s)

Sebastian Sippel

References

Martin, Plastino and Rosso (2006): Physica A 369 (2006) 439–462

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
opd = ordinal_pattern_distribution(x = x, ndemb = 6)
MPR_complexity(opd)
```

nbitflips — *A function to compute bitflip statistics and time series*

Description

Computation of bitflip statistics of a time series

Usage

```
nbitflips(x, ndemb)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the ordinal pattern distribution is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size) for which bitflips are to be calculated. Should be chosen such as length(x) » ndemb

Details

This function returns a histogram and time series of the number of bitflips occurring in the associated ordinal patterns. NA values are allowed, and any pattern that contains at least one NA value will be ignored. WARNING: Can be slow with very long time series (n > 10^7).

Value

A list with two entries is returned.

Author(s)

Sebastian Sippel

References

Sippel, S., Master Thesis, University of Bayreuth, 2014.

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^6)
nbitflips(x = x, ndemb = 6)
```

`ordinal_pattern_distribution` *A function to compute ordinal pattern statistics*

Description

Computation of the ordinal patterns of a time series (see e.g. Bandt and Pompe 2002)

Usage

```
ordinal_pattern_distribution(x, ndemb)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the ordinal pattern distribution is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb

Details

This function returns the distribution of ordinal patterns using the Keller coding scheme, detailed in Physica A 356 (2005) 114-120. NA values are allowed, and any pattern that contains at least one NA value will be ignored. (Fast) C routines are used for computing ordinal patterns.

Value

A character vector of length factorial(ndemb) is returned.

Author(s)

Sebastian Sippel

References

Bandt and Pompe, 2002.

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
ordinal_pattern_distribution(x = x, ndemb = 6)
```

`ordinal_pattern_distribution_2`
A function to compute ordinal pattern statistics

Description

Computation of the ordinal patterns of a time series (see e.g. Bandt and Pompe 2002)

Usage

```
ordinal_pattern_distribution(x, ndemb)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the ordinal pattern distribution is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb

Details

This function returns the distribution of ordinal patterns using the Keller coding scheme, detailed in Physica A 356 (2005) 114-120. NA values are allowed, and any pattern that contains at least on NA value will be ignored. The function uses old and slow R routines and is only maintained for comparability. For faster routines, see ordinal_pattern_distribution.

Value

A character vector of length factorial(ndemb) is returned.

Author(s)

Sebastian Sippel

References

Bandt and Pompe, 2002.

See Also

`ordinal_pattern_distribution`

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
ordinal_pattern_distribution(x = x, ndemb = 6)
```

`ordinal_pattern_time_series` *A function to compute time series of ordinal patterns*

Description

Computation of the ordinal patterns of a time series (see e.g. Bandt and Pompe 2002)

Usage

```
ordinal_pattern_time_series(x, ndemb)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the ordinal pattern time series is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb

Details

This function returns the distribution of ordinal patterns using the Keller coding scheme, detailed in Physica A 356 (2005) 114-120. NA values are allowed, and any pattern that contains at least one NA value will be ignored. (Fast) C routines are used for computing ordinal patterns.

Value

A character vector of length(x) is returned.

Author(s)

Sebastian Sippel

References

Bandt and Pompe, 2002.

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
ordinal_pattern_time_series(x = x, ndemb = 6)
```

`permutation_entropy` *A function to compute the permutation entropy*

Description

Computation of the permutation entropy of a time series based on its ordinal pattern distribution (see Bandt and Pompe 2002). Permutation entropy is a global information measure, hence insensitive to the permutation ordering scheme.

Usage

```
permutation_entropy(odp, x = NA, ndemb = NA, PatternCoding = "Default")
```

Arguments

`opd`	A numeric vector that details an ordinal pattern distribution.

Details

This function calculates the permutation entropy as described in Bandt and Pompe 2002.

Value

The normalized permutation entropy as a numeric value in the range [0;1].

Author(s)

Sebastian Sippel

References

Bandt and Pompe, 2002.

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
opd = ordinal_pattern_distribution(x = x, ndemb = 6)
permutation_entropy(opd)
```

`rank_to_permutation` *A function to convert a "ranks-based" permutation notation to an "index-based" permutation scheme.*

Description

Converts permutations denoted by ranks to permutations denoted by indices and back.

Usage

```
rank_to_permutation(target_pattern, permutation.notation)
```

Arguments

`pattern`	A numeric vector that denotes a permutation pattern.
`permutation.notation`	The permutation notation that should be used. Could be "Olivares.2012" or "Keller.2005".

Details

This function converts ranks to indices and back.

Value

A numeric vector, which contains the transformed permutation.

Author(s)

Sebastian Sippel

References

Sebastian Sippel (2014). Master Thesis. University of Bayreuth.

`transformPermCoding` *A function to generate a vector from an index-transformation vector from a permutation coding scheme*

Description

Generates a position vector to change the ordinal pattern distribution in the default permutation coding scheme (i.e. generated by ordinal_pattern_distribution(x, ndemb)) into a user-specified coding scheme. This is a required input for the function changePermCodingOPD.

Usage

```
transformPermCoding(target_pattern = "lehmerperm", ndemb = 4)
```

Arguments

`target_pattern`	A numeric matrix that specifies the pattern to be transformed into the position vector.
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb

Details

This function returns a character vector to transform the output of ordinal_pattern_distribution (permutation coding as of Keller and Sinn, 2005) into a user-specified permutation coding scheme. For example, pattern #5 in "lehmerperm" (ndemb = 5) is given by the ranks c(0, 1, 4, 2, 3). This corresponds to pattern #41 in the (original) Keller coding scheme, as given by transformPermCoding(target_pattern = "lehmerperm", ndemb = 5)[5].

Value

A numeric vector of length factorial(ndemb), which contains the positions of the corresponding patterns in the Keller Coding scheme.

Author(s)

Sebastian Sippel

References

see e.g. Olivares et al. 2012

Examples

```
transformPermCoding(target_pattern = "lehmerperm", ndemb = 4)
```

---

weighted_ordinal_pattern_distribution

A function to compute weighted ordinal pattern statistics

Description

Computation of weighted ordinal patterns of a time series. Weights can be generated by a user-specified function (e.g. variance-weighted, see Fadlallah et al 2013).

Usage

```
weighted_ordinal_pattern_distribution(x, ndemb)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the weighted ordinal pattern distribution is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb
`weight.fun`	Function to weight each pattern accordin to a user-specified function. This function must return one value. Default is to 'var.fun', which generates variance-based weights (see Fadlallah et al 2013).

Details

This function returns the distribution of weighted ordinal patterns using the Keller coding scheme, detailed in Physica A 356 (2005) 114-120. NA values are allowed. (Fast) C routines are used for computing weighted ordinal patterns.

Value

A character vector of length factorial(ndemb) is returned.

Author(s)

Sebastian Sippel

References

Fadlallah et al (2013). PHYSICAL REVIEW E 87, 022911 (2013)

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
weighted_ordinal_pattern_distribution(x = x, ndemb = 6)
```

---

weighted_ordinal_pattern_distribution_2

A function to compute weighted ordinal pattern statistics

Description

Computation of weighted ordinal patterns of a time series. Weights can be generated by a user-specified function (e.g. variance-weighted, see Fadlallah et al 2013).

Usage

```
weighted_ordinal_pattern_distribution(x, ndemb, weight.fun)
```

Arguments

`x`	A numeric vector (e.g. a time series), from which the weighted ordinal pattern distribution is to be calculated
`ndemb`	Embedding dimension of the ordinal patterns (i.e. sliding window size). Should be chosen such as length(x) » ndemb
`weight.fun`	Function to weight each pattern accordin to a user-specified function. This function must return one value. Default is to 'var.fun', which generates variance-based weights (see Fadlallah et al 2013).

Details

This function returns the distribution of weighted ordinal patterns using the Keller coding scheme, detailed in Physica A 356 (2005) 114-120. NA values are allowed. The function uses old and slow R routines and is only maintained for comparability. For faster routines, see weighted_ordinal_pattern_distribution.

Value

A character vector of length factorial(ndemb) is returned.

Author(s)

Sebastian Sippel

References

Fadlallah et al (2013). PHYSICAL REVIEW E 87, 022911 (2013)

See Also

`weighted_ordinal_pattern_distribution`

Examples

```
x = arima.sim(model=list(ar = 0.3), n = 10^4)
weighted_ordinal_pattern_distribution_2(x = x, ndemb = 6, weight.fun = var.fun)
```

Index