Package 'PatternCausality'

May 19, 2024

Type Package

Title Pattern Causality Algorithm

Version 0.1.0

Maintainer Hui Wang <hui.wang1@monash.edu>

Description The model proposes a robust methodology for detecting and reconstructing the hidden structure of dynamic complex systems through short-term forecasts and information embedded in reconstructed state spaces. It demonstrates significant accuracy in predicting interdependencies within systems by applying the method across diverse fields such as ecology, neurology, and finance, substantiating its efficacy with a 90% accuracy rate across over 100,000 simulations. The approach not only identifies critical components and causal interactions within these systems but also provides a practical tool for optimizing system performance and stability.

License GPL (>= 2)

Imports snow, snowfall, tcltk2, moments

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Author Stavros K. Stavroglou [aut, cre] (<https://orcid.org/0000-0003-3931-0391>), Athanasios Pantelous [aut] (<https://orcid.org/0000-0001-5738-1471>)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.1

Contents

`convertSignatureToValue`

Convert Signature to Predicted Values

Description

This function calculates predicted values of a series based on its previous value and predicted signature changes. It starts with an initial value from a historical series and sequentially adds predicted signature changes to generate a sequence of predicted values.

Usage

```
convertSignatureToValue(E, tau, Y, i, h, predictedSignatureY)
```

Arguments

`E`	Integer, the length of the series for which predictions are needed.
`tau`	Integer, the time delay used in the system dynamics (not used in this function but typically relevant in the broader context of time series prediction).
`Y`	Numeric vector, the original series from which the prediction starts.
`i`	Integer, the starting index in the vector Y from which the prediction should commence.
`h`	Integer, the horizon step for which the initial predicted value is directly obtained from Y.
`predictedSignatureY`	Numeric vector, the predicted changes (signature) at each step used for prediction.

Value

Numeric vector containing the predicted values of the series starting from index i+h in Y and extending for E steps, adjusted by the predicted signature.

Examples

```
Y <- c(1, 2, 3, 5, 8, 13, 21)
E <- 5
tau <- 1  # Example value, not used in the function
i <- 2
h <- 3
predictedSignatureY <- c(0.5, 1.5, 2.5, 3.5, 4.5)
predictedValues <- convertSignatureToValue(E, tau, Y, i, h, predictedSignatureY)
print(predictedValues)
```

`convertSignatureToValueOutOfSample`

Convert Out-of-Sample Signature to Predicted Values

Description

This function predicts future values for a time series starting from the last known predicted value. It uses a series of predicted signature changes to extrapolate future values, assuming that changes continue in a similar manner to the signatures provided. This is particularly useful in scenarios where short-term predictions are made in complex systems based on embedded dynamic structures as described in the associated literature.

Usage

```
convertSignatureToValueOutOfSample(
  E,
  tau,
  Y_pred_last,
  i,
  h,
  predictedSignatureY
)
```

Arguments

`E`	Integer, the number of future values to predict.
`tau`	Integer, the time delay used in the system dynamics; while not used directly in this function, it is relevant in the broader context of the methodology for embedding time series data.
`Y_pred_last`	Numeric, the last predicted value from which future values will be extrapolated.
`i`	Integer, the starting index for the prediction; not used in this function but generally important in the broader algorithm.
`h`	Integer, the horizon step from the last known actual value to the first prediction point; not used directly in this function but part of the overall predictive framework.
`predictedSignatureY`	Numeric vector, the predicted incremental changes (signature) at each step used for extrapolation of the series.

Value

Numeric vector containing the extrapolated future values of the series, starting from Y_pred_last and extending for E steps, adjusted by the predicted signature changes.

Examples

```
# Suppose Y_pred_last is the last known predicted value of a financial time series
Y_pred_last <- 120
# Assume predicted signature changes based on a model's output
predictedSignatureY <- c(2, 3, 4, 5, 6)
# Number of future points to predict
E <- 5
# Example values for tau and i, h not used in this function directly
tau <- 1
i <- 1
h <- 1

# Generate future predictions
futureValues <- convertSignatureToValueOutOfSample(E, tau, Y_pred_last, i, h, predictedSignatureY)
print(futureValues)
```

---

dataBank *Data Bank Initialization Function*

Description

Initializes various data structures for storing and managing data within a complex systems analysis framework.

Usage

```
dataBank(type, dimensions)
```

Arguments

type	A character string specifying the type of data structure to initialize: "array", "vector", "matrix", or "neighborhood memories". Each structure serves different requirements for data storage and processing in the model.
dimensions	An integer vector specifying the dimensions of the data structure. The interpretation of dimensions varies based on the type: • For "array", it defines the dimensions of the array. • For "vector", only the first element (length) is used. • For "matrix", the first two elements specify the number of rows and columns. • For "neighborhood memories", the first two elements define the number of rows and columns, the third element specifies the number of nearest neighbors, and the fourth element details the number of signature components per neighbor.

Value

db Returns the initialized data structure, filled with NA values. Depending on the 'type', the structure can be an array, vector, matrix, or a specialized data frame designed for neighborhood memories which incorporates extensive details about interactions within defined neighborhoods.

Examples

```
# Initialize a matrix with 3 rows and 5 columns.
matrix_db <- dataBank("matrix", c(3, 5))
print(matrix_db)

# Initialize a neighborhood memory structure correctly with sufficient column names.
dimensions_nm <- c(4, 40, 3, 5) # 4 rows, 40 columns, 3 neighbors, 5 signature components per neighbor
nm_db <- dataBank("neighborhood memories", dimensions_nm)
print(nm_db)
```

---

distanceMatrix *Compute Distance Matrix for an Embedded Matrix*

Description

This function computes the distance matrix from a given embedded matrix, `M`, using a specified metric. The distance matrix is essential for exploring the underlying structure in complex systems by quantifying the distances between different points (or states) in the embedded space. This matrix can be crucial for further analysis like clustering, nearest neighbor searches, or causality inference in complex systems.

Usage

```
distanceMatrix(M, metric)
```

Arguments

`M`	Numeric matrix, the embedded state space matrix where each row represents a point in the reconstructed state space of a time series or any multidimensional data.
`metric`	Character, the distance metric to be used for computing distances. Common metrics include "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".

Value

An object of class `dist`, representing the distance matrix of the embedded matrix `M`. This distance matrix can optionally be converted to a full matrix format if needed for subsequent analyses.

Examples

```
# Assume M is an already constructed state space matrix of a time series
M <- matrix(rnorm(100), nrow=10)
distanceMat <- distanceMatrix(M, "euclidean")
print(as.matrix(distanceMat))  # Optionally convert to a full matrix for display
```

`distanceVector` *Calculate Distances Between a Reference Point and Multiple Candidates*

Description

This function applies the 'metricDistance' function to calculate distances from a given reference point to each row in a matrix of candidate nearest neighbors. It is particularly useful in the situation between matrix and a vector

Usage

```
distanceVector(point, candidateNNs, n)
```

Arguments

`point`	Numeric vector, the reference point from which distances are calculated.
`candidateNNs`	Matrix, rows represent candidate points to which distance from the reference point is calculated.
`n`	Integer, the order of the Minkowski distance to use.

Value

Numeric vector, distances from the reference point to each of the candidate points.

Examples

```
point <- c(1, 2, 3)
candidateNNs <- matrix(c(4, 5, 6, 7, 8, 9), nrow=2, byrow=TRUE)
n <- 2
distances <- distanceVector(point, candidateNNs, n)
print(distances)
```

`firstCausalityPoint` *First Causality Point Function*

Description

Calculates the earliest index in a time series from which causality analysis can begin, based on the embedding dimension, time delay, prediction horizon, and the length of the series. It ensures that there are enough data points for conducting a causality analysis without running out of data.

Usage

```
firstCausalityPoint(E, tau, h, X)
```

Arguments

`E`	An integer representing the embedding dimension, which influences the number of dimensions in which the time series is reconstructed for analysis.
`tau`	An integer representing the time delay, used in reconstructing the time series in the embedded space.
`h`	An integer representing the prediction horizon, indicating how far ahead in the time series the predictions are aimed.
`X`	A numeric vector representing the time series data.

Value

An integer indicating the first index in the time series from which causality analysis is feasible without running out of data. If the parameters are set such that the analysis is not possible, the function will stop and provide an error message.

Examples

```
time_series <- rnorm(1000)  # Generate a random time series of 1000 points
embedding_dim <- 3  # Set embedding dimension
time_delay <- 2     # Set time delay
pred_horizon <- 1   # Set prediction horizon

# Calculate the first causality point
fc_point <- firstCausalityPoint(embedding_dim, time_delay, pred_horizon, time_series)
print(fc_point)
```

`firstCausalityPointCHECK` *First Causality Point Check Function*

Description

Checks if the time series data length is sufficient to perform causality analysis based on the provided embedding dimension, time delay, and prediction horizon. This function returns a Boolean indicating the feasibility of conducting the analysis.

Usage

```
firstCausalityPointCHECK(E, tau, h, X)
```

Arguments

`E`	An integer representing the embedding dimension, which influences the number of dimensions in which the time series is reconstructed for analysis.
`tau`	An integer representing the time delay, used in reconstructing the time series in the embedded space. Note that in this version of the function, 'tau' is not actively used in calculations.
`h`	An integer representing the prediction horizon, indicating how far ahead in the time series the predictions are aimed.
`X`	A numeric vector representing the time series data.

Value

A boolean value; 'TRUE' if the time series is long enough to accommodate the specified parameters without running out of data, 'FALSE' otherwise.

Examples

```
time_series <- rnorm(1000)  # Generate a random time series of 1000 points
embedding_dim <- 3  # Set embedding dimension
time_delay <- 2     # Set time delay (not used in current implementation)
pred_horizon <- 1   # Set prediction horizon

# Check if the first causality point can be considered
is_feasible <- firstCausalityPointCHECK(embedding_dim, time_delay, pred_horizon, time_series)
print(is_feasible)
```

metricDistance *Calculate Generalized Minkowski Distance Between Two Vectors*

Description

This function calculates the generalized Minkowski distance of order 'n' between two numeric vectors. This distance is a metric in a normed vector space which generalizes the Euclidean and Manhattan distances. It is used for various data science applications, particularly in clustering, optimization, and outlier detection in complex systems.

Usage

```
metricDistance(vec1, vec2, n)
```

Arguments

vec1	Numeric vector, the first vector for which the distance will be calculated.
vec2	Numeric vector, the second vector for which the distance will be calculated.
n	Integer, the order of the Minkowski distance. When n=2, it becomes the Euclidean distance; when n=1, it becomes the Manhattan distance.

Value

Numeric, the computed Minkowski distance between the two vectors.

Examples

```
vec1 <- c(1, 2, 3)
vec2 <- c(4, 5, 6)
n <- 2
distance <- metricDistance(vec1, vec2, n)
print(distance)
```

pastNNsInfo *Retrieve Information on Past Nearest Neighbors*

Description

This function extracts and organizes information about the nearest neighbors of a given point in time series data, utilizing past data up to a specified horizon. It operates within a framework that incorporates multiple matrices representing different attributes of the system's state space, such as distances, signatures, and patterns.

Usage

```
pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
```

Arguments

CCSPAN	Integer, the span of common coordinates to exclude in the nearest neighbor search.
NNSPAN	Integer, the number of nearest neighbors to consider for the analysis.
Mx	Matrix, the main matrix representing the state space of the system.
Dx	Numeric matrix, containing distances between points in the state space.
SMx	Matrix, containing signatures of the state space.
PSMx	Matrix, containing patterns derived from the signatures.
i	Integer, the current index in time series data for which nearest neighbors are being considered.
h	Integer, the horizon beyond which data is not considered in the nearest neighbor search.

Value

A list containing indices, times, distances, signatures, patterns, and original coordinates of the nearest neighbors from past data, useful for subsequent analysis and prediction in the context of complex systems.

Examples

```
# Random data generation for demonstration
set.seed(123)
Mx <- matrix(rnorm(200), nrow=20)
Dx <- as.matrix(dist(Mx))
SMx <- apply(Mx, 1, function(x) diff(x))
PSMx <- apply(SMx, 1, function(x) ifelse(x > 0, 1, ifelse(x < 0, -1, 0)))
CCSPAN <- 5
NNSPAN <- 3
i <- 15
h <- 2
neighborsInfo <- pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
print(neighborsInfo)
```

---

patternHashing *Pattern Hashing Function*

Description

This function hashes all possible patterns generated from a dataset to facilitate analysis of their distribution and frequency, supporting risk assessment in decision-making processes related to the causality and dynamics of complex systems.

Usage

```
patternHashing(E)
```

Arguments

E	The embedding dimension which influences the complexity and variety of patterns generated. This parameter adjusts the granularity with which the system's dynamics are analyzed and interpreted.

Value

hashedpatterns Returns a vector of hashed values representing each pattern or NA if the pattern generation was not possible, typically due to insufficient or overly simplified input.

Examples

```
# Assume E is set to 3, which is suitable for generating moderately complex patterns.
hashed_result <- patternHashing(3)
print(hashed_result)
```

---

patternSpace *Create Pattern Space from Signature Matrix*

Description

This function transforms a signature matrix into a pattern space matrix. Each row in the signature matrix is processed to reflect categorical changes (increase, decrease, no change) in the sequence values, which are then hashed to create unique pattern identifiers for further analysis. This transformation is crucial for identifying and categorizing patterns in complex systems, facilitating the exploration of underlying causal structures.

Usage

```
patternSpace(SM, E)
```

Arguments

SM	Matrix, the signature matrix where each row represents the differences between successive elements in the original time series data.
E	Integer, the number of dimensions in the signature matrix which influences the output size of the pattern space matrix.

Value

Matrix, where each row contains hashed pattern identifiers derived from the categorical changes in the signature matrix, facilitating pattern recognition and analysis in complex systems.

Examples

```
signatureMatrix <- matrix(c(1, -1, 0, 2, -2, 0, 1, -1, 0), nrow = 3, byrow = TRUE)
patternSpaceMatrix <- patternSpace(signatureMatrix, 3)
print(patternSpaceMatrix)
```

signatureSpace *Create Signature Space from Embedded Matrix*

Description

This function computes the signature space of a matrix obtained from a time series' state space by calculating the successive differences of each embedded vector. The signature space reveals changes between successive points in the time series, capturing dynamics that are crucial for understanding complex system behaviors.

Usage

```
signatureSpace(M, E)
```

Arguments

M	Matrix, the embedded state space matrix where each row is a point in the reconstructed state space of the time series.
E	Integer, the embedding dimension used to create the matrix M.

Value

Matrix where each row represents the vector of differences between successive elements of the corresponding row in matrix M. The orientation of the matrix may vary depending on the embedding dimension.

Examples

```
ts <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
E <- 3
tau <- 1
stateSpaceMatrix <- stateSpace(ts, E, tau)
signatureMatrix <- signatureSpace(stateSpaceMatrix, E)
print(signatureMatrix)
```

stateSpace *Construct State Space Matrix from Time Series*

Description

This function reconstructs the state space of a given time series by creating a delay embedding matrix. Each row in the matrix represents a point in the reconstructed state space, with embeddings based on specified lag (tau) and dimension (E). This approach is essential for analyzing the dynamics of complex systems, as it reveals underlying structures and patterns that are not immediately apparent in the raw time series data.

Usage

```
stateSpace(ts, E, tau)
```

Arguments

ts	Numeric vector, the original time series data from which the state space is to be reconstructed.
E	Integer, the embedding dimension, which determines the number of delayed copies of the time series to include in each row of the matrix.
tau	Integer, the time delay between each copy in the embedding, indicating the spacing in time steps between elements in each embedded vector.

Value

A matrix where each row corresponds to an embedded vector, representing a point in the reconstructed state space of the time series. Rows with insufficient data (due to the delay embedding reaching beyond the end of the time series) contain NA values.

Examples

```
ts <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
E <- 3
tau <- 1
stateSpaceMatrix <- stateSpace(ts, E, tau)
print(stateSpaceMatrix)
```

Index