Package 'patterncausality'

August 22, 2024

Type Package

Title Pattern Causality Algorithm

Version 0.1.2

Maintainer Hui Wang <huiw1128@gmail.com>

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. 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 MIT + file LICENSE

Depends R (>= 4.1.0)

Imports stats

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, stringr, ggplot2, testthat (>= 3.0.0)

Config/Needs/website tidyverse/tidytemplate

URL https://github.com/skstavroglou/pattern_causality/

BugReports https://github.com/skstavroglou/pattern_causality/issues

Config/testthat/edition 3

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Stavros Stavroglou [aut] (<https://orcid.org/0000-0003-3931-0391>), Athanasios Pantelous [aut] (<https://orcid.org/0000-0001-5738-1471>), Hui Wang [aut, cre] (<https://orcid.org/0009-0006-0095-0243>)

Repository CRAN

Date/Publication 2024-08-22 09:00:02 UTC

Contents

AUCO *A data from Illapel*

Description

Raw rodent and rainfall data from the study site at Las Chinchillas National Reserve near the city of Illapel, Coquimbo Region of Chil.

Usage

```
AUCO
```

Format

An object of class `data.frame` with 32 rows and 5 columns.

Examples

```
head(AUCO)
```

`climate` *Climate index*

Description

This is a time series of climate index.

Usage

```
climate
```

Format

An object of class `data.frame` with 535 rows and 5 columns.

Examples

```
head(climate)
```

`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)
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(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)
```

`fillPCMatrix` *Calculate and Record Causality Strengths*

Description

Computes the causality strengths based on the comparison between predicted and real patterns and signatures in a system's dynamic model. It applies a normalization function to measure the intensity of causal influences and uses an error function for weighting if required.

Usage

```
fillPCMatrix(
  weighted,
  predictedPatternY,
  realPatternY,
  predictedSignatureY,
  realSignatureY,
  patternX,
  signatureX
)
```

Arguments

`weighted`	Logical, if TRUE, the causality strength is calculated using the error function for normalization, otherwise a binary indication (1 for accurate prediction and 0 otherwise) is used.
`predictedPatternY`	Numeric vector, the predicted pattern of variable Y at a future time step.
`realPatternY`	Numeric vector, the actual observed pattern of variable Y at the same future time step.
`predictedSignatureY`	Numeric vector, the predicted signature vector derived from the system model for Y.
`realSignatureY`	Numeric vector, the actual observed signature vector for Y.
`patternX`	Numeric vector, the current observed pattern of variable X, used as the basis for prediction.
`signatureX`	Numeric vector, the current observed signature vector of variable X.

Details

Fill Pattern Causality Matrix with Causality Strengths

This function calculates and fills the causality strengths between predicted and real patterns and signatures for a complex system analysis. It evaluates the accuracy of predictions and computes the strength of causal relationships based on norm vectors and optionally weights these strengths using the error function (erf).

Value

A dataframe with two columns: 'real' and 'predicted', representing the real and predicted causality strengths.

Examples

```
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
My <- matrix(rnorm(200), nrow = 20)
Dx <- distanceMatrix(Mx, "minkowski")
Dy <- distanceMatrix(My, "minkowski")
SMx <- signatureSpace(Mx, E)
SMy <- signatureSpace(My, E)
PSMx <- patternSpace(SMx, E)
PSMy <- patternSpace(SMy, E)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
NNx <- pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
timesX <- NNx$times
projNNy <- projectedNNsInfo(My, Dy, SMy, PSMy, timesX, i, h)
pSY <- predictionY(E, projNNy, zeroTolerance = E - 1)$predictedSignatureY
pPY <- predictionY(E, projNNy, zeroTolerance = E - 1)$predictedPatternY[1]
rSY <- SMy[(i + h), ]
rPY <- PSMy[i + h]
signatureX <- SMx[i, ]
patternX <- PSMx[i, ]
weighted <- 0
pc <- fillPCMatrix(weighted, pPY, rPY, pSY, rSY, patternX, signatureX)
```

---

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)
```

---

natureOfCausality *Evaluate the Nature of Causality in Dynamic Complex Systems*

Description

This function analyzes a three-dimensional pattern causality matrix to classify the nature of causality (positive, negative, dark, or no causality) between pairs of variables across specified time points. It is designed to interpret the dynamics within complex systems by examining the causal relationships encoded in the matrix.

Usage

```
natureOfCausality(PC, dur, hashedpatterns, X)
```

Arguments

PC	A three-dimensional array where each slice along the third dimension represents a pattern causality matrix at a specific time point, encoding the strength and type of causality between pairs of variables.
dur	A numeric vector indicating the time points (slices of the PC matrix) to analyze for causality.
hashedpatterns	A numeric vector of hashed or indexed pattern identifiers that correspond to variables in the system, used for interpreting matrix dimensions in causality checks.
X	An auxiliary numeric vector used to determine the length of the output vectors for causality results, typically aligning with the number of time points or variables.

Value

A data frame with four columns: 'noCausality', 'Positive', 'Negative', and 'Dark', each containing binary indicators (1 for presence, 0 for absence) that correspond to the presence of each causality type at each time point analyzed.

Examples

```
# Generate a sample 3D causality matrix with random data
set.seed(123) # For reproducibility
PC <- array(runif(300), dim = c(10, 10, 3)) # 10x10 matrix over 3 time points
dur <- 1:3 # Time points to analyze
hashedpatterns <- seq(1, 10) # Simulated hashed pattern identifiers
X <- rep(0, 3) # Auxiliary vector for output length

# Run the natureOfCausality function
results <- natureOfCausality(PC, dur, hashedpatterns, X)
print(results)
```

---

optimalParametersSearch

Optimal Parameters Search for Causality Analysis

Description

Searches for the optimal embedding dimension (E) and time delay (tau) to maximize the accuracy of causality predictions in a dataset. It evaluates each combination of E and tau for their ability to predict different types of causality: total, positive, negative, and dark.

Usage

```
optimalParametersSearch(Emax, tauMax, metric, dataset)
```

Arguments

Emax	The maximum embedding dimension to test.
tauMax	The maximum time delay to test.
metric	The distance metric to use in the causality analysis (e.g., 'euclidean').
dataset	A matrix where each column represents a time series to be analyzed.

Value

A data frame summarizing the causality analysis results across all tested E and tau values, showing the mean total, positive, negative, and dark causality accuracies for each parameter combination.

Examples

```
data(climate)
dataset <- climate[, -1]

optimalParams <- optimalParametersSearch(Emax=3, tauMax=3, metric="euclidean", dataset=dataset)
print(optimalParams)
```

pastNNsInfo *Finding Nearest Neighbors and Keeping their Topological Information*

Description

This function identifies the nearest neighbors of a given point in a time series, excluding common coordinate vectors and a specified horizon from the candidate nearest neighbors. It returns detailed information about these neighbors, including their times, distances, signatures, patterns, and coordinates.

Usage

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

Arguments

CCSPAN	Integer, the span of common coordinates to exclude from 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:

- i: The current index in time series data.
- times: The times of the nearest neighbors.
- dists: The distances to the nearest neighbors.
- signatures: The signatures of the nearest neighbors.
- patterns: The patterns of the nearest neighbors.
- coordinates: The coordinates of the nearest neighbors.

Examples

```
# Generate random data for demonstration
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
Dx <- distanceMatrix(Mx, "minkowski")
SMx <- signatureSpace(Mx, E)
PSMx <- patternSpace(SMx, E)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
neighborsInfo <- pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
print(neighborsInfo)
```

`pastNNsInfo_Lite` *Finding Nearest Neighbors and Keeping their Topological Information (Lite Version)*

Description

This function is a simplified version of `pastNNsInfo`, identifying the nearest neighbors of a given point in a time series. It returns detailed information about these neighbors, including their times, distances, signatures, patterns, and coordinates, without excluding common coordinate vectors and horizon.

Usage

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

Arguments

`CCSPAN`	Integer, the span of common coordinates to exclude from the nearest neighbor search (not used in Lite version).
`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 (not used in Lite version).

Value

A list containing:

- `i`: The current index in time series data.
- `times`: The times of the nearest neighbors.
- `dists`: The distances to the nearest neighbors.
- `signatures`: The signatures of the nearest neighbors.
- `patterns`: The patterns of the nearest neighbors.
- `coordinates`: The coordinates of the nearest neighbors.

Examples

```
# Generate random data for demonstration
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
Dx <- distanceVector(point = Mx[i, ], candidateNNs = Mx[1:(i - CCSPAN - h), ], n = 2)
SMx <- signatureSpace(Mx, E)
PSMx <- patternSpace(SMx, E)
neighborsInfoLite <- pastNNsInfo_Lite(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
print(neighborsInfoLite)
```

---

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)
```

pcAccuracy *PC Accuracy Evaluation Function*

Description

This function evaluates the causality prediction accuracy across multiple time series within a dataset using the PC Mk. II Light method. It checks the feasibility of causality analysis between all pairs, computes different types of causality (total, positive, negative, dark), and aggregates these results.

Usage

```
pcAccuracy(dataset, E, tau, metric, h, weighted)
```

Arguments

dataset	A matrix or data frame where each column represents a time series.
E	The embedding dimension used for state space reconstruction.
tau	The time delay used in state space reconstruction.
metric	A character string specifying the distance metric to be used (e.g., 'euclidean', 'maximum').
h	Prediction horizon, indicating how far ahead in the time series the predictions are aimed.
weighted	A logical indicating whether to use a weighted approach in calculating the causality strengths.

Value

A data frame with the embedding parameters and average values of total, positive, negative, and dark causality across all time series pairs in the dataset.

Examples

```
data(climate)
data <- climate[, -1]
results <- pcAccuracy(data, E = 3, tau = 1, metric = "euclidean", h = 1, weighted = TRUE)
print(results)
```

pcFullDetails *Pattern Causality Mark II: Full Details*

Description

Implements an advanced pattern causality algorithm to explore the causal relationships between two time series datasets. This function reconstructs state spaces, calculates distances, and evaluates causality using predefined metrics and pattern analysis. The methodology supports complex system analysis where traditional linear methods fall short.

Usage

```
pcFullDetails(X, Y, E, tau, metric, h, weighted)
```

Arguments

X	Numeric vector, the first time series data.
Y	Numeric vector, the second time series data.
E	Integer, the embedding dimension used in the reconstruction of the state space.
tau	Integer, the time delay between data points in the reconstruction.
metric	Character, the distance metric used (e.g., 'euclidean', 'manhattan').
h	Integer, the prediction horizon, specifying the number of steps ahead for causality analysis.
weighted	Logical, specifies if causality strength should be weighted.

Value

A list containing various outputs including matrices of nearest neighbors, predicted and actual causality matrices, signature matrices, pattern vectors, and diagnostics related to the time series analysis and causality assessment.

Examples

```
data(climate)
X <- climate$AO
Y <- climate$AAO
result <- pcFullDetails(X, Y, E = 3, tau = 2, metric = "euclidean", h = 1, weighted = TRUE)
print(result)
```

pcLightweight *Pattern Causality Lightweight Function*

Description

This function implements the Pattern Causality Model Mk. II for lightweight analysis of causal interactions between two time series using pattern and signature spaces. It assesses causality through reconstructed state spaces and hashed pattern analysis.

Usage

```
pcLightweight(X, Y, E, tau, metric, h, weighted, tpb = TRUE)
```

Arguments

X	A numeric vector representing the first time series.
Y	A numeric vector representing the second time series.
E	The embedding dimension, which influences the number of dimensions in which the time series is reconstructed for analysis.
tau	The time delay used in reconstructing the time series in the embedded space.
metric	A character string indicating the distance metric to be used (e.g., 'euclidean', 'maximum').
h	The prediction horizon, representing the number of steps ahead for which predictions are needed.
weighted	A logical indicating whether to use a weighted approach in the causality strength calculations.
tpb	A bool parameter to show progress bar.

Value

A data frame with columns for total, positive, negative, and dark causality percentages across evaluated time points, giving insights into the nature of causality between the time series.

Examples

```
data(climate)
X <- climate$AO
Y <- climate$AAO
result <- pcLightweight(X, Y, E = 3, tau = 1, metric = "euclidean", h = 2, weighted = TRUE)
print(result)
```

predictionY *Predict Signature and Pattern Vectors*

Description

Uses neural network outputs to predict the state signatures and patterns in a complex system. Adjusts for sparsity using zero tolerance.

Usage

```
predictionY(E, projNNy, zeroTolerance = (E + 1)/2)
```

Arguments

E	Integer, the embedding dimension of the system, indicating the length of the signature vector minus one.
projNNy	A list containing two elements: Signatures, a matrix where each column represents a component in the signature vector across different observations, and Weights, a numeric vector representing the weights associated with each observation.
zeroTolerance	A numeric value used to determine the sparsity threshold in the signature matrix. Default is set to (E+1)/2.

Details

Predict the signatures and patterns for a complex system

This function predicts the signature and pattern vectors for a given state based on projections and weights derived from neural network outputs within a complex system. The predictions adjust according to a specified zero tolerance level to manage sparsity.

Value

A dataframe with two columns: predictedSignatureY which contains the predicted signature vector, and predictedPatternY which contains the corresponding pattern vector.

Examples

```
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
My <- matrix(rnorm(200), nrow = 20)
Dx <- distanceMatrix(Mx, "minkowski")
Dy <- distanceMatrix(My, "minkowski")
SMx <- signatureSpace(Mx, E)
SMy <- signatureSpace(My, E)
PSMx <- patternSpace(SMx, E)
```

```
PSMy <- patternSpace(SMy, E)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
NNx <- pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
timesX <- NNx$times
projNNy <- projectedNNsInfo(My, Dy, SMy, PSMy, timesX, i, h)
predicted <- predictionY(E, projNNy)
print(predicted)
```

`projectedNNsInfo` *Projected Nearest Neighbors Information*

Description

Extracts and returns information about the projected nearest neighbors in a time series context, specifically useful for understanding interactions in dynamic complex systems.

Usage

```
projectedNNsInfo(My, Dy, SMy, PSMy, timesX, i, h)
```

Arguments

`My`	Matrix of coordinates in the original space.
`Dy`	Distance matrix, representing distances between elements in My.
`SMy`	Matrix of signatures, capturing essential patterns in the data.
`PSMy`	Matrix of patterns, representing characteristic configurations of the data.
`timesX`	Index at which the projection starts.
`i`	Index of the specific element for which information is being extracted.
`h`	Horizon over which the projection is considered.

Value

A list containing indices of the element, the projected times, distances, weights derived from these distances, signatures, patterns, and the coordinates of the nearest neighbors.

Examples

```
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
My <- matrix(rnorm(200), nrow = 20)
Dx <- distanceMatrix(Mx, "minkowski")
Dy <- distanceMatrix(My, "minkowski")
```

```
SMx <- signatureSpace(Mx, E)
SMy <- signatureSpace(My, E)
PSMx <- patternSpace(SMx, E)
PSMy <- patternSpace(SMy, E)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
NNx <- pastNNsInfo(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
timesX <- NNx$times
projNNy <- projectedNNsInfo(My, Dy, SMy, PSMy, timesX, i, h)
print(projNNy)
```

`projectedNNsInfo_Lite` *Projected Nearest Neighbors Information (Lite Version)*

Description

Simplified version of `projectedNNsInfo` function, intended for contexts where the distance matrix is directly passed as a parameter, applicable in scenarios where distances are precomputed or not dependent on dynamic indices.

Usage

```
projectedNNsInfo_Lite(My, Dy, SMy, PSMy, timesX, i, h)
```

Arguments

`My`	Matrix of coordinates in the original space.
`Dy`	Precomputed distance matrix for all elements.
`SMy`	Matrix of signatures.
`PSMy`	Matrix of patterns.
`timesX`	Index at which the projection starts.
`i`	Index of the specific element for which information is being extracted.
`h`	Horizon over which the projection is considered.

Value

A list similar to `projectedNNsInfo`, containing details about the element, projected times, distances, weights, signatures, patterns, and coordinates.

Examples

```
set.seed(123)
E <- 3
tau <- 1
Mx <- matrix(rnorm(200), nrow = 20)
My <- matrix(rnorm(200), nrow = 20)
CCSPAN <- (E - 1) * tau
NNSPAN <- E + 1
i <- 15
h <- 2
Dx <- distanceVector(point = Mx[i, ], candidateNNs = Mx[1:(i - CCSPAN - h), ], n = 2)
Dy <- distanceVector(point = My[i, ], candidateNNs = My[1:(i - CCSPAN - h), ], n = 2)
SMx <- signatureSpace(Mx, E)
SMy <- signatureSpace(My, E)
PSMx <- patternSpace(SMx, E)
PSMy <- patternSpace(SMy, E)
NNx <- pastNNsInfo_Lite(CCSPAN, NNSPAN, Mx, Dx, SMx, PSMx, i, h)
timesX <- NNx$times
projNNy <- projectedNNsInfo_Lite(My, Dy, SMy, PSMy, timesX, i, h)
print(projNNy)
```

`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

```
data(climate)
ts <- climate$AO
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)
```

stock *Stock index*

Description

This is a time series of stock series which contains AAPL and MSFT from March 13, 1986 to August 6, 2018.

Usage

```
stock
```

Format

An object of class `data.frame` with 8167 rows and 12 columns.

Examples

```
head(stock)
```

Index