Package ‘rEDM’

October 16, 2021

Type Package

Title Empirical Dynamic Modeling (‘EDM’)

Version 1.9.3

Date 2021-09-25

Maintainer Joseph Park <JosephPark@IEEE.org>

Description An implementation of 'EDM' algorithms based on research software developed for internal use at the Sugihara Lab ('UCSD/SIO'). The package is implemented with 'Rcpp' wrappers around the 'cppEDM' library. It implements the 'simplex' projection method from Sugihara & May (1990) <doi:10.1038/344734a0>, the 'S-map' algorithm from Sugihara (1994) <doi:10.1098/rsta.1994.0106>, convergent cross mapping described in Sugihara et al. (2012) <doi:10.1126/science.1227079>, and, 'multiview embedding' described in Ye & Sugihara (2016) <doi:10.1126/science.aag0863>.

License BSD_2_clause + file LICENSE

LazyData true

LazyLoad yes

Imports methods, Rcpp (>= 1.0.1)

LinkingTo Rcpp, RcppThread

Suggests knitr, rmarkdown, formatR

VignetteBuilder knitr

NeedsCompilation yes

Author Joseph Park [aut, cre] (<https://orcid.org/0000-0001-5411-1409>), Cameron Smith [aut] (<https://orcid.org/0000-0003-0020-5607>), George Sugihara [aut, cct] (<https://orcid.org/0000-0002-2863-6946>), Ethan Deyle [aut] (<https://orcid.org/0000-0001-8704-8434>), Erik Saberski [ctb] (<https://orcid.org/0000-0002-6475-6187>), Hao Ye [ctb] (<https://orcid.org/0000-0002-8630-1458>), The Regents of the University of California [cph]

Repository CRAN

Date/Publication 2021-10-16 08:20:02 UTC
**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_3sp</td>
<td>2</td>
</tr>
<tr>
<td>block_gp</td>
<td>3</td>
</tr>
<tr>
<td>block_lnlp</td>
<td>4</td>
</tr>
<tr>
<td>CCM</td>
<td>7</td>
</tr>
<tr>
<td>ccm</td>
<td>8</td>
</tr>
<tr>
<td>circle</td>
<td>11</td>
</tr>
<tr>
<td>ComputeError</td>
<td>11</td>
</tr>
<tr>
<td>compute_stats</td>
<td>12</td>
</tr>
<tr>
<td>EDM</td>
<td>12</td>
</tr>
<tr>
<td>Embed</td>
<td>14</td>
</tr>
<tr>
<td>EmbedDimension</td>
<td>15</td>
</tr>
<tr>
<td>EvergladesFlow</td>
<td>16</td>
</tr>
<tr>
<td>Lorenz5D</td>
<td>16</td>
</tr>
<tr>
<td>MakeBlock</td>
<td>17</td>
</tr>
<tr>
<td>make_block</td>
<td>18</td>
</tr>
<tr>
<td>make_surrogate_data</td>
<td>19</td>
</tr>
<tr>
<td>Multiview</td>
<td>19</td>
</tr>
<tr>
<td>multiview</td>
<td>21</td>
</tr>
<tr>
<td>paramecium_didinium</td>
<td>23</td>
</tr>
<tr>
<td>PredictInterval</td>
<td>24</td>
</tr>
<tr>
<td>PredictNonlinear</td>
<td>25</td>
</tr>
<tr>
<td>sardine_anchovy_sst</td>
<td>26</td>
</tr>
<tr>
<td>Simplex</td>
<td>27</td>
</tr>
<tr>
<td>simplex</td>
<td>28</td>
</tr>
<tr>
<td>SMap</td>
<td>31</td>
</tr>
<tr>
<td>SurrogateData</td>
<td>33</td>
</tr>
<tr>
<td>TentMap</td>
<td>34</td>
</tr>
<tr>
<td>TentMapNoise</td>
<td>35</td>
</tr>
<tr>
<td>Thrips</td>
<td>35</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_3sp</td>
<td>36</td>
</tr>
</tbody>
</table>

---

**Description**

Time series generated from a discrete-time coupled Lotka-Volterra model exhibiting chaotic dynamics.

**Usage**

`block_3sp`
Format

A data frame with 198 rows and 10 columns:

- time: time index (# of generations)
- $x_{t}$: abundance of simulated species $x$ at time $t$
- $x_{t-1}$: abundance of simulated species $x$ at time $t-1$
- $x_{t-2}$: abundance of simulated species $x$ at time $t-2$
- $y_{t}$: abundance of simulated species $y$ at time $t$
- $y_{t-1}$: abundance of simulated species $y$ at time $t-1$
- $y_{t-2}$: abundance of simulated species $y$ at time $t-2$
- $z_{t}$: abundance of simulated species $z$ at time $t$
- $z_{t-1}$: abundance of simulated species $z$ at time $t-1$
- $z_{t-2}$: abundance of simulated species $z$ at time $t-2$

Arguments

- block: not implemented
- lib: not implemented
- pred: not implemented
- tp: not implemented
- phi: not implemented
- v_e: not implemented
**block_lnlp**

Perform generalized forecasting using simplex projection or s-map

---

**Description**

block_lnlp uses multiple time series given as input to generate an attractor reconstruction, and then applies the simplex projection or s-map algorithm to make forecasts. This method generalizes the simplex and s_map routines, and allows for "mixed" embeddings, where multiple time series can be used as different dimensions of an attractor reconstruction.

**Usage**

```r
block_lnlp(block, lib = NULL, pred = NULL, norm = 2, method = c("simplex", "s-map"), tp = 1, num_neighbors = switch(match.arg(method), simplex = "e+1", `s-map` = 0), columns = NULL, target_column = 1, stats_only = TRUE, first_column_time = FALSE, exclusion_radius = NULL, epsilon = NULL, theta = NULL, silent = TRUE, save_smap_coefficients = FALSE)
```

---

**Value**

Not implemented.
Arguments

block | either a vector to be used as the time series, or a data.frame or matrix where each column is a time series
lib | a 2-column matrix, data.frame, 2-element vector or string of row indice pairs, where each pair specifies the first and last rows of the time series to create the library. If not specified, all available rows are used
pred | (same format as lib), but specifying the sections of the time series to forecast. If not specified, set equal to lib
norm | the distance measure to use. see 'Details'
method | the prediction method to use. see 'Details'
num_neighbors | the number of nearest neighbors to use. Note that the default value will change depending on the method selected. (any of "e+1", "E+1", "e + 1", "E + 1" will set this parameter to E+1 for each run.)
columns | either a vector with the columns to use (indices or names), or a list of such columns
target_column | the index (or name) of the column to forecast
stats_only | specify whether to output just the forecast statistics or to include the raw predictions for each run
first_column_time | indicates whether the first column of the given block is a time column (and therefore excluded when building the library)
exclusion_radius | excludes vectors from the search space of nearest neighbors if their time index is within exclusion_radius (NULL turns this option off)
epsilon | Not implemented
theta | the nonlinear tuning parameter (theta is only relevant if method == "s-map")
silent | prevents warning messages from being printed to the R console
save_smap_coefficients | specifies whether to include the s_map coefficients with the output

Details

The default parameters are set so that passing a vector as the only argument will use that vector to predict itself one time step ahead. If a matrix or data.frame is given as the only argument, the first column will be predicted (one time step ahead), using the remaining columns as the embedding. If the first column is not a time vector, 1:NROW will be used as time values.
norm = 2 (only option currently available) uses the "L2 norm", Euclidean distance:

\[
\text{distance}(a,b) := \sqrt{\sum_i (a_i - b_i)^2}
\]

method "simplex" (default) uses the simplex projection forecasting algorithm
method "s-map" uses the s-map forecasting algorithm
Value

A data.frame with components for the parameters and forecast statistics:

- `cols`  
  - embedding

- `tp`  
  - prediction horizon

- `nn`  
  - number of neighbors

- `num_pred`  
  - number of predictions

- `rho`  
  - correlation coefficient between observations and predictions

- `mae`  
  - mean absolute error

- `rmse`  
  - root mean square error

- `perc`  
  - percent correct sign

- `p_val`  
  - p-value that rho is significantly greater than 0 using Fisher's z-transformation

- `const_pred_rho`  
  - same as rho, but for the constant predictor

- `const_pred_mae`  
  - same as mae, but for the constant predictor

- `const_pred_rmse`  
  - same as rmse, but for the constant predictor

- `const_pred_perc`  
  - same as perc, but for the constant predictor

- `const_p_val`  
  - same as p_val, but for the constant predictor

- `model_output`  
  - data.frame with columns for the time index, observations, predictions, and estimated prediction variance (if `stats_only == FALSE`)

If "s-map" is the method, then the same, but with additional columns:

- `theta`  
  - the nonlinear tuning parameter

- `smap_coefficients`  
  - data.frame with columns for the s-map coefficients (if `save_smap_coefficients == TRUE`)

- `smap_coefficient_covariances`  
  - list of covariance matrices for the s-map coefficients (if `save_smap_coefficients == TRUE`)

Examples

```r
block <- block_3sp
block_lnlp(block[,2:4])

block <- block_3sp
block_lnlp(block[,1:4], first_column_time = TRUE)

block <- block_3sp
block_lnlp(block, target_column = "x_t", columns = c("y_t", "z_t"), first_column_time = TRUE)

block <- block_3sp
x_t_pred = block_lnlp(block, columns = c("x_t", "y_t"), first_column_time = TRUE, stats_only = FALSE)

block <- block_3sp
x_t_pred = block_lnlp(block, method = "s-map", theta = 3, columns =
c("x_t", "y_t"), first_column_time = TRUE, stats_only = FALSE, save_smap_coefficients = TRUE)
```
Convergent cross mapping using simplex projection

Description

The state-space of a multivariate dynamical system (not a purely stochastic one) encodes coherent phase-space variable trajectories. If enough information is available, one can infer the presence or absence of cross-variable interactions associated with causal links between variables. CCM measures the extent to which states of variable Y can reliably estimate states of variable X. This happens only if X is causally influencing Y.

If cross-variable state predictability converges as more state-space information is provided, this indicates a causal link. CCM performs this cross-variable mapping using Simplex, with convergence assessed across a range of observational library sizes as described in Sugihara et al. 2012.

Usage

CCM(pathIn = "./", dataFile = "", dataFrame = NULL, pathOut = "./", predictFile = "", E = 0, Tp = 0, knn = 0, tau = -1, exclusionRadius = 0, columns = "", target = ", libSizes = ", sample = 0, random = TRUE, replacement = FALSE, seed = 0, includeData = FALSE, verbose = FALSE, showPlot = FALSE)

Arguments

pathIn path to dataFile.
dataFile .csv format data file name. The first column must be a time index or time values. The first row must be column names.
dataFrame input data.frame. The first column must be a time index or time values. The columns must be named.
pathOut path for predictFile containing output predictions.
predictFile output file name.
E embedding dimension.
Tp prediction horizon (number of time column rows).
knn number of nearest neighbors. If knn=0, knn is set to E+1.
tau lag of time delay embedding specified as number of time column rows.
exclusionRadius excludes vectors from the search space of nearest neighbors if their relative time index is within exclusionRadius.
columns string of whitespace separated column name(s) in the input data used to create the library.
target column name in the input data used for prediction.
libSizes string of 3 whitespace separated integer values specifying the initial library size, the final library size, and the library size increment.
sample  integer specifying the number of random samples to draw at each library size evaluation.
random  logical to specify random (TRUE) or sequential library sampling.
replacement  logical to specify sampling with replacement.
seed  integer specifying the random sampler seed. If seed=0 then a random seed is generated.
includeData  logical to include statistics and predictions for every prediction in the ensemble.
verbose  logical to produce additional console reporting.
showPlot  logical to plot results.

Details

CCM computes the X:Y and Y:X cross-mappings in parallel using threads.

Value

A data.frame with 3 columns. The first column is LibSize specifying the subsampled library size. Columns 2 and 3 report Pearson correlation coefficients for the prediction of X from Y, and Y from X.

References


Examples

data(sardine_anchovy_sst)
df <- CCM( dataFrame=sardine_anchovy_sst, E=3, Tp=0, columns="anchovy", target="np_sst", libSizes="10 70 10", sample=100 )

Description

ccm uses time delay embedding on one time series to generate an attractor reconstruction, and then applies the simplex projection algorithm to estimate concurrent values of another time series. This method is typically applied, varying the library sizes, to determine if one time series contains the necessary dynamic information to recover the influence of another, causal variable.
Usage

ccm(block, lib = NULL, pred = NULL, norm = 2, E = 1, tau = -1,
    tp = 0, num_neighbors = "e+1", lib_sizes = c(10, 75, 5),
    random_libs = TRUE, num_samples = 100, replace = FALSE, lib_column = 1,
    target_column = 2, first_column_time = FALSE, RNGseed = NULL,
    exclusion_radius = NULL, epsilon = NULL, stats_only = TRUE,
    silent = TRUE)

Arguments

block: either a vector to be used as the time series, or a data.frame or matrix where each column is a time series

lib: a 2-column matrix, data.frame, 2-element vector or string of row indice pairs, where each pair specifies the first and last *rows* of the time series to create the library. If not specified, all available rows are used

pred: (same format as lib), but specifying the sections of the time series to forecast. If not specified, set equal to lib

norm: the distance measure to use. see 'Details'

E: the embedding dimensions to use for time delay embedding

tau: the time-delay offset to use for time delay embedding

tp: the prediction horizon (how far ahead to forecast)

num_neighbors: the number of nearest neighbors to use. Note that the default value will change depending on the method selected. (any of "e+1", "E+1", "e + 1", "E + 1" will set this parameter to E+1 for each run

lib_sizes: three integers specifying the start, stop and increment index of library sizes

random_libs: indicates whether to use randomly sampled libs

num_samples: is the number of random samples at each lib size (this parameter is ignored if random_libs is FALSE)

replace: indicates whether to sample vectors with replacement

lib_column: name (index) of the column to cross map from

target_column: name (index) of the column to forecast

first_column_time: indicates whether the first column of the given block is a time column

RNGseed: will set a seed for the random number generator, enabling reproducible runs of ccm with randomly generated libraries

exclusion_radius: excludes vectors from the search space of nearest neighbors if their *time index* is within exclusion_radius (NULL turns this option off)

epsilon: not implemented

stats_only: specify whether to output just the forecast statistics or the raw predictions for each run

silent: prevents warning messages from being printed to the R console
Details

ccm runs both forward and reverse cross maps in separate threads. Results are returned for both mappings. The default parameters are set so that passing a matrix as the only argument will use \( E = 1 \) (embedding dimension), and leave-one-out cross-validation over the whole time series to compute cross-mapping from the first column to the second column, letting the library size vary from 10 to 75 in increments of 5.

\( \text{norm} = 2 \) (only option currently available) uses the "L2 norm", Euclidean distance:

\[
\text{distance}(a, b) := \sqrt{\sum_{i} (a_i - b_i)^2}
\]

Value

If \( \text{stats\_only} = \text{TRUE} \): a data.frame with forecast statistics for both the forward and reverse mappings:

- LibSize library length (number of vectors)
- x:y cross mapped correlation coefficient between observations x and predictions y
- y:x cross mapped correlation coefficient between observations y and predictions x
- E embedding dimension
- tau time delay offset
- tp forecast interval
- nn number nearest neighbors

If \( \text{stats\_only} = \text{FALSE} \): a named list with the following items: settings:

- LibMeans data.frame with the mean bidirectional forecast statistics
- CCM1_PredictStat data.frame with forward mapped prediction statistics for each prediction of the ensemble
- CCM1_Predictions list of prediction result data.frame each forward mapped prediction of the ensemble
- CCM2_PredictStat data.frame with reverse mapped prediction statistics for each prediction of the ensemble
- CCM2_Predictions list of prediction result data.frame each reverse mapped prediction of the ensemble

CCM1_PredictStat and CCM2_PredictStat data.frames have columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>prediction number</td>
</tr>
<tr>
<td>E</td>
<td>embedding dimension</td>
</tr>
<tr>
<td>nn</td>
<td>number of nearest neighbors</td>
</tr>
<tr>
<td>tau</td>
<td>embedding time delay offset</td>
</tr>
<tr>
<td>LibSize</td>
<td>library size</td>
</tr>
<tr>
<td>rho</td>
<td>correlation coefficient</td>
</tr>
<tr>
<td>RMSE</td>
<td>root mean square error</td>
</tr>
<tr>
<td>MAE</td>
<td>maximum absolute error</td>
</tr>
<tr>
<td>lib</td>
<td>column name of the library vector</td>
</tr>
<tr>
<td>target</td>
<td>column name of the target vector</td>
</tr>
</tbody>
</table>
Examples

```r
anchovy_xmap_sst <- ccm(sardine_anchovy_sst, E = 3,
  lib_column = "anchovy", target_column = "np_sst",
  lib_sizes = c(10, 75, 5), num_samples = 100)
```

---

**circle**  

*2-D timeseries of a circle.*

---

**Description**

Time series of a circle in 2-D ($\sin$ and $\cos$).

**Usage**

`circle`

**Format**

A data frame with 200 rows and 3 columns:

- **Time**: time index.
- **x**: $\sin$ component.
- **y**: $\cos$ component.

---

**ComputeError**  

*Compute error*

---

**Description**

`ComputeError` evaluates the Pearson correlation coefficient, mean absolute error and root mean square error between two numeric vectors.

**Usage**

`ComputeError(obs, pred)`

**Arguments**

- **obs**: vector of observations.
- **pred**: vector of predictions.

**Value**

A name list with components:
**Examples**

```r
data(block_3sp)
smplx <- Simplex(dataFrame=block_3sp, lib="1 99", pred="105 190", E=3,
columns="x_t", target="x_t")
err <- ComputeError(smplx$Observations, smplx$Predictions)
```

**compute_stats**

*Compute performance metrics for predictions*

**Description**

Computes the rho, MAE, RMSE, perc, and p-val performance metrics.

**Arguments**

- `observed`: a vector of the observed values
- `predicted`: a vector of corresponding predicted values

**Value**

A data.frame with components with various performance metrics:

- `num_pred`: number of predictions
- `rho`: correlation coefficient between observations and predictions
- `mae`: mean absolute error
- `rmse`: root mean square error
- `perc`: percent correct sign
- `p_val`: p-value that rho is significantly greater than 0 using Fisher’s

**Examples**

```r
compute_stats(rnorm(100), rnorm(100))
```

EDM

*Empirical dynamic modeling*

**Description**

EDM provides tools for data-driven time series analyses. It is based on reconstructing multivariate state (or phase) space representations from uni or multivariate time series, then projecting state changes using various metrics applied to nearest neighbors.
EDM is a Rcpp interface to the cppEDM library of Empirical Dynamic Modeling tools. Function-
ality includes:

- Simplex projection (Sugihara and May 1990)
- Sequential Locally Weighted Global Linear Maps (S-map) (Sugihara 1994)
- Multivariate embeddings (Dixon et. al. 1999)
- Convergent cross mapping (Sugihara et. al. 2012)
- Multiview embedding (Ye and Sugihara 2016)

Details

Main Functions:

- **Simplex** - simplex projection
- **SMap** - S-map projection
- **CCM** - convergent cross mapping
- **Multiview** - multiview forecasting

Helper Functions:

- **Embed** - time delay embedding
- **ComputeError** - forecast skill metrics
- **EmbedDimension** - optimal embedding dimension
- **PredictInterval** - optimal prediction interval
- **PredictNonlinear** - evaluate nonlinearity

Author(s)

**Maintainer**: Joseph Park & Cameron Smith

**Authors**: Joseph Park, Cameron Smith, Ethan Deyle, Erik Saberski, George Sugihara

References

Sugihara G. and May R. 1990. Nonlinear forecasting as a way of distinguishing chaos from mea-

Transactions: Physical Sciences and Engineering, 348 (1688) : 477-495.

283:1528-1530.


Ye H., and G. Sugihara, 2016. Information leverage in interconnected ecosystems: Overcoming the
Embed data with time lags

Description

Embed performs Takens time-delay embedding on columns.

Usage

Embed(path = "/", dataFile = "", dataFrame = NULL, E = 0, tau = -1, columns = "", verbose = FALSE)

Arguments

- `path`: path to dataFile.
- `dataFile`: .csv format data file name. The first column must be a time index or time values. The first row must be column names. One of dataFile or dataFrame are required.
- `dataFrame`: input data.frame. The first column must be a time index or time values. The columns must be named. One of dataFile or dataFrame are required.
- `E`: embedding dimension.
- `tau`: integer time delay embedding lag specified as number of time column rows.
- `columns`: string of whitespace separated column name(s) in the input data to be embedded.
- `verbose`: logical to produce additional console reporting.

Details

Each columns item will have E-1 time-lagged vectors created. The column name is appended with (t-n). For example, data columns X, Y, with E = 2 will have columns named X(t-0) X(t-1) Y(t-0) Y(t-1).

The returned data.frame does not have a time column. The returned data.frame is truncated by tau * (E-1) rows to remove state vectors with partial data (NaN elements).

Value

A data.frame with lagged columns. E columns for each variable specified in columns.

Examples

data(circle)
embed <- Embed( dataFrame = circle, E = 2, tau = -1, columns = "x y" )
EmbedDimension

Optimal embedding dimension

Description

EmbedDimension uses Simplex to evaluate prediction accuracy as a function of embedding dimension.

Usage

EmbedDimension(pathIn = "/", dataFile = "", dataFrame = NULL, pathOut = "", predictFile = "", lib = "", pred = "", maxE = 10, Tp = 1, tau = -1, columns = "", target = "", embedded = FALSE, verbose = FALSE, numThreads = 4, showPlot = TRUE)

Arguments

pathIn  path to dataFile.
dataFile .csv format data file name. The first column must be a time index or time values. The first row must be column names.
dataFrame input data.frame. The first column must be a time index or time values. The columns must be named.
pathOut path for predictFile containing output predictions.
predictFile output file name.
lib string with start and stop indices of input data rows used to create the library of observations. A single contiguous range is supported.
pred string with start and stop indices of input data rows used for predictions. A single contiguous range is supported.
maxE maximum value of E to evaluate.
Tp prediction horizon (number of time column rows).
tau lag of time delay embedding specified as number of time column rows.
columns string of whitespace separated column name(s) in the input data used to create the library.
target column name in the input data used for prediction.
embedded logical specifying if the input data are embedded.
verbose logical to produce additional console reporting.
numThreads number of parallel threads for computation.
showPlot logical to plot results.

Value

A data.frame with columns E, rho.
Examples

data(TentMap)
E.rho <- EmbedDimension( dataFrame=TentMap, lib="1 100", pred="201 500",
columns="TentMap", target="TentMap", showPlot=FALSE)

EvergladesFlow  Water flow to NE Everglades

Description

Cumulative weekly water flow into northeast Everglades from water control structures S12C, S12D and S333 from 1980 through 2005.

Usage

EvergladesFlow

Format

A data frame with 1379 rows and 2 columns:

Date Date.
S12CD_S333_CFS  Cumulative weekly flow (CFS).

Lorenz5D  5-D Lorenz’96

Description

5-D Lorenz’96 timeseries with F = 8.

Usage

Lorenz5D

Format

Data frame with 1000 rows and 6 columns

Time Time.
V1 variable 1.
V2 variable 2.
V3 variable 3.
V4 variable 4.
V5 variable 5.
MakeBlock

References

MakeBlock Make embedded data block

Description
MakeBlock performs Takens time-delay embedding on columns. It is an internal function called by Embed that does not perform input error checking or validation.

Usage
MakeBlock(dataFrame, E = 0, tau = -1, columns = "", deletePartial = FALSE)

Arguments
dataFrame input data.frame. The first column must be a time index or time values. The columns must be named.
E embedding dimension.
tau integer time delay embedding lag specified as number of time column rows.
columns string of whitespace separated column name(s) in the input data to be embedded.
deletePartial boolean to delete rows with partial data.

Details
Each columns item will have E-1 time-lagged vectors created. The column name is appended with (t-n). For example, data columns X, Y, with E = 2 will have columns named X(t-0) X(t-1) Y(t-0) Y(t-1).

The returned data.frame does not have a time column.
If deletePartial is TRUE, the returned data.frame is truncated by tau * (E-1) rows to remove state vectors with partial data (NaN elements).

Value
A data.frame with lagged columns. E columns for each variable specified in columns.

Examples
data(TentMap)
embed <- MakeBlock(TentMap, 3, 1, "TentMap")
**make_block**

*Make a time delay offset block*

**Description**

`make_block` generates a time offset block with the appropriate `max_lag` and `tau`. The first column is presumed to be a time or index vector, and is not included in the embedding.

**Usage**

```r
make_block(block, columns = NULL, t = NULL, max_lag = 3, tau = -1, lib = NULL, restrict_to_lib = TRUE)
```

**Arguments**

- `block`: a data.frame or matrix where each column is a time series
- `columns`: list of column names to time delay.
- `t`: Not used
- `max_lag`: the total number of lags to include for each variable. So if `max_lag == 3`, a variable `X` is offset with lags `X[t]`, `X[t + tau]`, `X[t + 2*tau]`
- `tau`: the time delay offset for embedding
- `lib`: not used
- `restrict_to_lib`: not used

**Value**

A data.frame with time offset columns. If the original block had columns X, Y and `max_lag = 3`, then the returned data.frame will have columns `X(t-0) X(t-1) X(t-2) Y(t-0) Y(t-1) Y(t-2)`.

**Examples**

```r
data("block_3sp")
make_block(block_3sp[, c(1, 2, 5)])
```
make_surrogate_data

Generate surrogate data for permutation/randomization tests

Description

This is a wrapper function for generating surrogate time series using several different null models.

Usage

make_surrogate_data(ts, method = c("random_shuffle", "ebisuzaki", "seasonal"), num_surr = 100, T_period = 1, alpha = 0)

Arguments

ts the original time series
method which algorithm to use to generate surrogate data
num_surr the number of null surrogates to generate
T_period the period of seasonality for seasonal surrogates (ignored for other methods)
alpha standard deviation of seasonal cycle deviates.

Value

A matrix where each column is a separate surrogate with the same length as 'ts'.

Examples

data = make_surrogate_data(block_3sp$x_t)

Multiview

Forecasting using multiview embedding

Description

Multiview applies the method of Ye & Sugihara to find optimal combinations of variables that best represent the dynamics.

Usage

Multiview(pathIn = "./", dataFile = "", dataFrame = NULL, pathOut = "./", predictFile = "", lib = "", pred = "", D = 0, E = 1, Tp = 1, knn = 0, tau = -1, columns = "", target = "", multiview = 0, exclusionRadius = 0, trainLib = TRUE, excludeTarget = FALSE, verbose = FALSE, numThreads = 4, showPlot = FALSE)
Arguments

pathIn  path to dataFile.
dataFile  .csv format data file name. The first column must be a time index or time values. The first row must be column names.
dataFrame  input data.frame. The first column must be a time index or time values. The columns must be named.
pathOut  path for predictFile containing output predictions.
predictFile  prediction output file name.
lib  a 2-column matrix, data.frame, 2-element vector or string of row indice pairs, where each pair specifies the first and last *rows* of the time series to create the library.
pred  (same format as lib), but specifying the sections of the time series to forecast.
D  multivariate dimension.
E  embedding dimension.
Tp  prediction horizon (number of time column rows).
knn  number of nearest neighbors. If knn=0, knn is set to E+1.
tau  lag of time delay embedding specified as number of time column rows.
columns  string of whitespace separated column name(s) in the input data used to create multivariable data sets.
target  column name in the input data used for prediction.
multiview  number of multiview ensembles to average for the final prediction estimate.
exclusionRadius  number of adjacent observation vector rows to exclude as nearest neighbors in prediction.
trainLib  logical to use in-sample (lib=pred) projections for the ranking of column combinations.
excludeTarget  logical to exclude embedded target column from combinations.
verbose  logical to produce additional console reporting.
numThreads  number of CPU threads to use in multiview processing.
showPlot  logical to plot results.

Details

Multiview embedding is a method to identify variables in a multivariate dynamical system that are most likely to contribute to the observed dynamics. It is a multistep algorithm with these general steps:

1. Compute D-dimensional variable combination forecasts.
2. Rank forecasts.
3. Compute predictions of top combinations.
4. Compute multiview averaged prediction.
If $E > 1$, all variables are embedded to dimension $E$. If `trainLib` is `TRUE` initial forecasts and ranking are done in-sample (lib=pred) and predictions using the top ranked combinations use the specified lib and pred. If `trainLib` is `FALSE` initial forecasts and ranking use the specified lib and pred, the step of computing predictions of the top combinations is skipped.

**Value**

Named list with data.frames `[[Combo_rho,Predictions]]`.

- **data.frame** `Combo_rho` columns:
  - Col_1 column index
  - ... column index
  - Col_E column index
  - rho Pearson correlation
  - MAE mean absolute error
  - RMSE root mean square error

**References**


**Examples**

```r
data(block_3sp)
L = Multiview(dataFrame = block_3sp, lib = "1 99", pred = "105 190",
            E = 2, columns = "x_t y_t z_t", target = "x_t")
```

---

**Description**

`multiview` applies the method described in Ye & Sugihara (2016) for forecasting, where multiple attractor reconstructions are tested, and a single nearest neighbor is selected from each of the top $k$ reconstructions to produce final forecasts.

**Usage**

```r
multiview(block, lib = NULL, pred = NULL, norm = 2, E = 1, tau = -1,
         tp = 1, max_lag = 3, num_neighbors = "e+1", k = "sqrt", na.rm = FALSE,
         target_column = 1, stats_only = TRUE, save_lagged_block = FALSE,
         first_column_time = FALSE, exclusion_radius = NULL, silent = FALSE)
```
Arguments

block
either a vector to be used as the time series, or a data.frame or matrix where each
column is a time series

lib
a 2-column matrix, data.frame, 2-element vector or string of row indice pairs,
where each pair specifies the first and last *rows* of the time series to create the
library. If not specified, all available rows are used

pred
(same format as lib), but specifying the sections of the time series to forecast. If
not specified, set equal to lib

norm
the distance measure to use. see ’Details’

E
the embedding dimensions to use for time delay embedding. The default value
of 1 does not embed the data.

tau
the time-delay offset to use for time delay embedding

tp
the prediction horizon (how far ahead to forecast)

max_lag
the maximum number of lags to use for variable combinations. If max_lag ==
3, a variable X will be embedded with lags X[t], X[t + tau], X[t + 2*tau]

num_neighbors
the number of nearest neighbors to use. Note that the default value will change
depending on the method selected. (any of ”e+1”, ”E+1”, ”e + 1”, ”E + 1” will
set this parameter to E+1 for each run.)

k
the number of embeddings to use for ensemble averaging. ”sqrt” or 0 will use k
= sqrt(m) where m is the number of multiview combinations of the set of input
variables

na.rm
logical. Should missing values (including ’NaN” be omitted from the calcula-
tions?)

target_column
the name (index) of the column to forecast

stats_only
specify whether to output just the forecast statistics or the raw predictions for
each run

save_lagged_block
specify whether to output the lagged block that is constructed as part of running
multiview

first_column_time
indicates whether the first column of the given block is a time column and ex-
cluded when building the library

exclusion_radius
excludes vectors from the search space of nearest neighbors if their *time index*
is within exclusion_radius (NULL turns this option off)

silent
prevents warning messages from being printed to the R console

Details

*multiview* uses multiple time series given as input to generate an attractor reconstruction, and then
applies the simplex projection to make forecasts. This method generalizes the *simplex* routine, and
allows for ”mixed” embeddings, where multiple time series can be used as different dimensions of
an attractor reconstruction.
paramecium_didinium

The default parameters are set so that, given a matrix of time series, forecasts will be produced for the first column. By default, all possible combinations of the columns are used for the attractor construction, the \( k = \sqrt{m} \) heuristic will be used, forecasts will be one time step ahead. If a time vector is not supplied, 1:NROW will be used. The default lib and pred are to use the first half of the data for the "library" and to predict over the second half of the data. Unless otherwise set, the output will be just the forecast statistics.

\( \text{norm} = 2 \) (only option currently available) uses the "L2 norm", Euclidean distance:

\[
distance(a, b) := \sqrt{\sum_i (a_i - b_i)^2}
\]

Value

A named list with items "View" and "Predictions". View is a data.frame with components:

- \( \text{col}_i, \ldots, \text{col}_j \) column indices of the embedding
- \( \text{name}_i, \ldots, \text{nam}_j \) column names of the embedding
- \( \rho \) correlation of the projection
- \( \text{MAE} \) maximum absolute error of the projection
- \( \text{RMSE} \) root mean square error of the projection

Predictions is a data.frame of the predictions from the best multiview ensemble.

Examples

```r
block <- block_3sp[, c(2, 5, 8)]
multiview( block, k=10 )
```

---

paramecium_didinium  Time series for the Paramecium-Didinium laboratory experiment

Description

Time series of Paramecium and Didinium abundances (\#/mL) from an experiment by Veilleux (1979)

Usage

paramecium_didinium
**PredictInterval**  
*Forecast interval accuracy*

### Description

*PredictInterval* uses **Simplex** to evaluate prediction accuracy as a function of forecast interval $T_p$.

### Usage

```r
PredictInterval(pathIn = "./", dataFile = "", dataFrame = NULL, pathOut = "./",  
predictFile = "", lib = "", pred = "", maxTp = 10, E = 1, tau = -1,  
columns = "", target = ",", embedded = FALSE, verbose = FALSE,  
numThreads = 4, showPlot = TRUE)
```

### Arguments

- **pathIn**: path to `dataFile`
- **dataFile**: .csv format data file name. The first column must be a time index or time values. The first row must be column names.
- **dataFrame**: input data.frame. The first column must be a time index or time values. The columns must be named.
- **pathOut**: path for `predictFile` containing output predictions.
- **predictFile**: output file name.
- **lib**: string with start and stop indices of input data rows used to create the library of observations. A single contiguous range is supported.
- **pred**: string with start and stop indices of input data rows used for predictions. A single contiguous range is supported.
- **maxTp**: maximum value of $T_p$ to evaluate.
- **E**: embedding dimension.
- **tau**: lag of time delay embedding specified as number of time column rows.
- **columns**: string of whitespace separated column name(s) in the input data used to create the library.
- **target**: column name in the input data used for prediction.
- **embedded**: logical specifying if the input data are embedded.
- **verbose**: logical to produce additional console reporting.
- **numThreads**: number of parallel threads for computation.
- **showPlot**: logical to plot results.

### Value

A data.frame with columns $T_p$, $\rho$. 
Examples

data(TentMap)
Tp.rho <- PredictInterval( dataFrame=TentMap, lib="1 100",
    pred="201 500", E=2, columns="TentMap", target="TentMap", showPlot = FALSE)

PredictNonlinear Test for nonlinear dynamics

Description

PredictNonlinear uses SMap to evaluate prediction accuracy as a function of the localisation parameter theta.

Usage

PredictNonlinear(pathIn = "./", dataFile = "", dataFrame = NULL,
    pathOut = "./", predictFile = "", lib = "", pred = "", theta = "",
    E = 1, Tp = 1, knn = 0, tau = -1, columns = "", target = "",
    embedded = FALSE, verbose = FALSE, numThreads = 4, showPlot = TRUE)

Arguments

pathIn path to dataFile.
dataFile .csv format data file name. The first column must be a time index or time values. The first row must be column names.
dataFrame input data.frame. The first column must be a time index or time values. The columns must be named.
pathOut path for predictFile containing output predictions.
predictFile output file name.
lib string with start and stop indices of input data rows used to create the library of observations. A single contiguous range is supported.
pred string with start and stop indices of input data rows used for predictions. A single contiguous range is supported.
theta A whitespace delimited string with values of the S-map localisation parameter. An empty string will use default values of [0.01 0.1 0.3 0.5 0.75 1 1.5 2 3 4 5 6 7 8 9].
E embedding dimension.
Tp prediction horizon (number of time column rows).
knn number of nearest neighbors. If knn=0, knn is set to the library size.
tau lag of time delay embedding specified as number of time column rows.
columns string of whitespace separated column name(s) in the input data used to create the library.
target column name in the input data used for prediction.
embedded logical specifying if the input data are embedded.
verbose logical to produce additional console reporting.
umThreads number of parallel threads for computation.
showPlot logical to plot results.

Details

The localisation parameter theta weights nearest neighbors according to \( \exp\left(-\theta \frac{D}{D_{avg}}\right) \) where \( D \) is the distance between the observation vector and neighbor, \( D_{avg} \) the mean distance. If \( \theta = 0 \), weights are uniformly unity corresponding to a global autoregressive model. As \( \theta \) increases, neighbors in closer proximity to the observation are considered.

Value

A data.frame with columns Theta, rho.

Examples

data(TentMapNoise)
theta.rho <- PredictNonlinear( dataFrame=TentMapNoise, E=2,lib="1 100", pred="201 500", columns="TentMap", target="TentMap", showPlot = FALSE)

data

sardine_anchovy_sst  Time series for the California Current Anchovy-Sardine-SST system

Description

Time series of Pacific sardine landings (CA), Northern anchovy landings (CA), and sea-surface temperature (3-year average) at the SIO pier and Newport pier

Usage

sardine_anchovy_sst

Format

year year of measurement
anchovy anchovy landings, scaled to mean = 0, sd = 1
sardine sardine landings, scaled to mean = 0, sd = 1
sio_sst 3-year running average of sea surface temperature at SIO pier, scaled to mean = 0, sd = 1
np_sst 3-year running average of sea surface temperature at Newport pier, scaled to mean = 0, sd = 1
Simplex forecasting

Description

Simplex performs time series forecasting based on weighted nearest neighbors projection in the time series phase space as described in Sugihara and May.

Usage

Simplex(pathIn = "./", dataFile = "", dataFrame = NULL, pathOut = "./", predictFile = "", lib = "", pred = "", E = 0, Tp = 1, knn = 0, tau = -1, exclusionRadius = 0, columns = "", target = "", embedded = FALSE, verbose = FALSE, const_pred = FALSE, validLib = vector(), showPlot = FALSE)

Arguments

pathIn path to dataFile.
dataFile .csv format data file name. The first column must be a time index or time values. The first row must be column names.
dataFrame input data.frame. The first column must be a time index or time values. The columns must be named.
pathOut path for predictFile containing output predictions.
predictFile output file name.
lib string with start and stop indices of input data rows used to create the library of observations. A single contiguous range is supported.
pred string with start and stop indices of input data rows used for predictions. A single contiguous range is supported.
E embedding dimension.
Tp prediction horizon (number of time column rows).
knn number of nearest neighbors. If knn=0, knn is set to E+1.
tau lag of time delay embedding specified as number of time column rows.
exclusionRadius excludes vectors from the search space of nearest neighbors if their relative time index is within exclusionRadius.
columns string of whitespace separated column name(s) in the input data used to create the library.
target column name in the input data used for prediction.
embedded logical specifying if the input data are embedded.
verbose logical to produce additional console reporting.
const_pred logical to add a constant predictor column to the output. The constant predictor is \( X(t+1) = X(t) \).

validLib logical vector the same length as the number of data rows. Any data row represented in this vector as FALSE, will not be included in the library.

showPlot logical to plot results.

Details

If embedded is FALSE, the data column(s) are embedded to dimension \( E \) with time lag \( \tau \). This embedding forms an \( E \)-dimensional phase space for the Simplex projection. If embedded is TRUE, the data are assumed to contain an \( E \)-dimensional embedding with \( E \) equal to the number of columns. Predictions are made using leave-one-out cross-validation, i.e. observation vectors are excluded from the prediction simplex.

To assess an optimal embedding dimension \( \text{EmbedDimension} \) can be applied. Accuracy statistics can be estimated by \( \text{ComputeError} \).

Value

A data.frame with columns Observations, Predictions. If const_pred is TRUE the column Const_Predictions is added. The first column contains the time values.

References


Examples

data(block_3sp)
smplx <- Simplex( dataFrame=block_3sp, lib="1 99", pred="105 190", E=3, columns="x_t", target="x_t" )
ComputeError(smplx$Predictions, smplx$Observations)

simplex

Perform univariate forecasting

Description

\texttt{simplex} uses time delay embedding on a single time series to generate an attractor reconstruction, and then applies the simplex projection algorithm to make forecasts.

\texttt{s_map} is similar to \texttt{simplex}, but uses the S-map algorithm to make forecasts.
Usage

```r
details = getDetails()
simplex(time_series, lib = NULL, pred = NULL, norm = 2, E = 1:10,
tau = -1, tp = 1, num_neighbors = "e+1", stats_only = TRUE,
exclusion_radius = NULL, epsilon = NULL, silent = TRUE)

ds_map(time_series, lib = NULL, pred = NULL, norm = 2, E = 1,
tau = -1, tp = 1, num_neighbors = 0, theta = NULL, stats_only = TRUE,
exclusion_radius = NULL, epsilon = NULL, silent = TRUE,
save_smap_coefficients = FALSE)
```

Arguments

time_series either a vector to be used as the time series, or a data.frame or matrix with at least 2 columns (in which case the first column will be used as the time index, and the second column as the time series)

lib a 2-column matrix, data.frame, 2-element vector or string of row indice pairs, where each pair specifies the first and last *rows* of the time series to create the library. If not specified, all available rows are used

pred (same format as lib), but specifying the sections of the time series to forecast. If not specified, set equal to lib

norm the distance measure to use. see 'Details'

E the embedding dimensions to use for time delay embedding

tau the time-delay offset to use for time delay embedding

tp the prediction horizon (how far ahead to forecast)

num_neighbors the number of nearest neighbors to use. Note that the default value will change depending on the method selected. (any of "e+1", "E+1", "e + 1", "E + 1" will set this parameter to E+1.)

stats_only specify whether to output just the forecast statistics or the raw predictions for each run

exclusion_radius excludes vectors from the search space of nearest neighbors if their *time index* is within exclusion_radius (NULL turns this option off)

epsilon Deprecated.

silent prevents warning messages from being printed to the R console

theta the nonlinear tuning parameter (theta is only relevant if method == "s-map")

save_smap_coefficients specifies whether to include the s_map coefficients with the output

Details

`simplex` is typically applied, and the embedding dimension varied, to find an optimal embedding dimension for the data. Thus, the default parameters are set so that passing a time series as the only argument will run over E = 1:10 (embedding dimension), using leave-one-out cross-validation over the whole time series, and returning just the forecast statistics.
**s_map** is typically applied, with fixed embedding dimension, and theta varied, to test for nonlinear dynamics in the data. Thus, the default parameters are set so that passing a time series as the only argument will run over a default list of thetas (0, 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 0.5, 0.75, 1.0, 1.5, 2, 3, 4, 6, and 8), using \( E = 1 \), leave-one-out cross-validation over the whole time series, and returning just the forecast statistics.

\[ \text{norm} = 2 \] (only option currently available) uses the "L2 norm", Euclidean distance:

\[
distance(a, b) := \sqrt{\sum_i (a_i - b_i)^2}
\]

**Value**

For **simplex**, if `stats_only = TRUE`: a data.frame with components for the parameters and forecast statistics:

- \( E \) embedding dimension
- \( \tau \) embedding time offset
- \( \text{tp} \) prediction horizon
- \( \text{nn} \) number of neighbors
- \( \text{num_pred} \) number of predictions
- \( \rho \) correlation coefficient between observations and predictions
- \( \text{mae} \) mean absolute error
- \( \text{rmse} \) root mean square error
- \( \text{perc} \) percent correct sign
- \( \text{p_val} \) p-value that \( \rho \) is significantly greater than 0 using Fisher’s z-transformation
- \( \text{const_pred_rho} \) same as \( \rho \), but for the constant predictor
- \( \text{const_pred_mae} \) same as \( \text{mae} \), but for the constant predictor
- \( \text{const_pred_rmse} \) same as \( \text{rmse} \), but for the constant predictor
- \( \text{const_pred_perc} \) same as \( \text{perc} \), but for the constant predictor
- \( \text{const_p_val} \) same as \( \text{p_val} \), but for the constant predictor

For **simplex**, if `stats_only = FALSE`: a named list with data.frame "stats" specified above, and named list "model_output":

- \( \text{model_output} \) named list with data.frames for each model. Columns include the time index, observations, predictions, and estimated prediction variance

For **s_map**, if `stats_only = TRUE`, the same data.frame as for **simplex**, but with additional column:

- \( \theta \) the nonlinear tuning parameter

For **s_map**, if `save_smap_coefficients = TRUE`, a named list with data.frame "stats" specified above and the following list items:

- \( \text{smap_coefficients} \) data.frame with columns for the \( s \)-map coefficients
- \( \text{smap_coefficient_covariances} \) list of covariance matrices for the \( s \)-map coefficients
For `s_map`, if `stats_only = FALSE`, a named list with data.frame "stats" specified above, and named list "model_output":

- **model_output** named list with data.frames for each model. Columns include the time index, observations, predictions, and estimated prediction variance.

### Examples

```r
ts <- block_3sp$x_t
simplex(ts, lib = c(1, 100), pred = c(101, 190))

ts <- block_3sp$x_t
simplex(ts, stats_only = FALSE)

ts <- block_3sp$x_t
s_map(ts, E = 2)

ts <- block_3sp$x_t
s_map(ts, E = 2, theta = 1, save_smap_coefficients = TRUE)
```

---

**SMap forecasting**

**Description**

SMap performs time series forecasting based on localised (or global) nearest neighbor projection in the time series phase space as described in Sugihara 1994.

**Usage**

```r
SMap(pathIn = './', dataFile = '', dataFrame = NULL, pathOut = './', predictFile = '', lib = '', pred = '', E = 0, Tp = 1, knn = 0, tau = -1, theta = 0, exclusionRadius = 0, columns = '', target = '', smapFile = '', jacobians = '', embedded = FALSE, const_pred = FALSE, verbose = FALSE, validLib = vector(), showPlot = FALSE)
```

**Arguments**

- `pathIn` path to `dataFile`.
- `dataFile` .csv format data file name. The first column must be a time index or time values. The first row must be column names.
- `dataFrame` input data.frame. The first column must be a time index or time values. The columns must be named.
- `pathOut` path for `predictFile` containing output predictions.
- `predictFile` prediction output file name.
lib          string with start and stop indices of input data rows used to create the library of observations. A single contiguous range is supported.
pred         string with start and stop indices of input data rows used for predictions. A single contiguous range is supported.
E            embedding dimension.
Tp            prediction horizon (number of time column rows).
knn          number of nearest neighbors. If knn=0, knn is set to the library size.
tau          lag of time delay embedding specified as number of time column rows.
theta         neighbor localisation exponent.
exclusionRadius          excludes vectors from the search space of nearest neighbors if their relative time index is within exclusionRadius.
columns       string of whitespace separated column name(s) in the input data used to create the library.
target        column name in the input data used for prediction.
smapFile      output file containing SMap coefficients.
jacobians     not used.
embedded      logical specifying if the input data are embedded.
const_pred    logical to add a constant predictor column to the output. The constant predictor is \(X(t+1) = X(t)\).
verbose       logical to produce additional console reporting.
validLib      logical vector the same length as the number of data rows. Any data row represented in this vector as FALSE, will not be included in the library.
showPlot      logical to plot results.

Details

If embedded is FALSE, the data column(s) are embedded to dimension E with time lag tau. This embedding forms an n-columns * E-dimensional phase space for the SMap projection. If embedded is TRUE, the data are assumed to contain an E-dimensional embedding with E equal to the number of columns. See the Note below for proper use of multivariate data (number of columns > 1).

Predictions are made using leave-one-out cross-validation, i.e. observation rows are excluded from the prediction regression.

In contrast to Simplex, SMap uses all available neighbors and weights them with an exponential decay in phase space distance with exponent theta. theta=0 uses all neighbors corresponding to a global autoregressive model. As theta increases, neighbors closer in vicinity to the observation are considered.

Value

A named list with two data.frames [[predictions,coefficients]]. predictions has columns Observations, Predictions. If const_pred is TRUE the column Const_Predictions is added. The first column contains time values.

coefficients data.frame has time values in the first column. Columns 2 through E+2 (E+1 columns) are the SMap coefficients.
Note

SMap should be called with columns explicitly corresponding to dimensions E. In the univariate case (number of columns = 1) with default embedded = FALSE, the time series will be time-delay embedded to dimension E, SMap coefficients correspond to each dimension.

If a multivariate data set is used (number of columns > 1) it must use embedded = TRUE with E equal to the number of columns. This prevents the function from internally time-delay embedding the multiple columns to dimension E. If the internal time-delay embedding is performed, then state-space columns will not correspond to the intended dimensions in the matrix inversion, coefficient assignment, and prediction. In the multivariate case, the user should first prepare the embedding (using Embed for time-delay embedding), then pass this embedding to SMap with appropriately specified columns, E, and embedded = TRUE.

References


Examples

data(circle)
L = SMap( dataFrame=circle,lib="1 100", pred="110 190", theta=4, E=2, embedded=TRUE,columns="x y", target="x")

SurrogateData

Generate surrogate data for permutation/randomization tests

Description

SurrogateData generates surrogate data under several different null models.

Usage

SurrogateData( ts, method = c("random_shuffle", "ebisuza", "seasonal"), num_surr = 100, T_period = 1, alpha = 0 )

Arguments

ts the original time series
method which algorithm to use to generate surrogate data
num_surr the number of null surrogates to generate
T_period the period of seasonality for seasonal surrogates (ignored for other methods)
alpha additive noise factor: N(0, alpha)
Details

Method "random_shuffle" creates surrogates by randomly permuting the values of the original time series.

Method “Ebisuzaki” creates surrogates by randomizing the phases of a Fourier transform, preserving the power spectra of the null surrogates.

Method "seasonal" creates surrogates by computing a mean seasonal trend of the specified period and shuffling the residuals. It is presumed that the seasonal trend can be extracted with a smoothing spline. Additive Gaussian noise is included according to N(0, alpha).

Value

A matrix where each column is a separate surrogate with the same length as ts.

Examples

data("block_3sp")
ts <- block_3sp$x_t
SurrogateData(ts, method = "ebisuzaki")

TentMap

Time series for a tent map with mu = 2.

Description

First-differenced time series generated from the tent map recurrence relation with mu = 2.

Usage

TentMap

Format

Data frame with 999 rows and 2 columns

Time  time index.
TentMap  tent map values.
**TentMapNoise**

| TentMapNoise | Time series of tent map plus noise. |

**Description**
First-differenced time series generated from the tent map recurrence relation with \( \mu = 2 \) and random noise.

**Usage**
TentMapNoise

**Format**
Data frame with 999 rows and 2 columns
- Time  time index.
- TentMap  tent map values.

---

**Thrips**

| Thrips | Apple-blossom Thrips time series |

**Description**
Seasonal outbreaks of Thrips imaginis.

**References**
Index

* datasets
  - block_3sp, 2
  - circle, 11
  - EvergladesFlow, 16
  - Lorenz5D, 16
  - paramecium_didinium, 23
  - sardine_anchovy_sst, 26
  - TentMap, 34
  - TentMapNoise, 35

* package
  - EDM, 12

  - block_3sp, 2
  - block_gp, 3
  - block_lnlp, 4, 4

  - CCM, 7, 7, 8, 13
  - ccm, 8, 8, 10
  - ccm_means (block_gp), 3
  - circle, 11
  - compute_stats, 12
  - ComputeError, 11, 11, 13, 28

  - EDM, 12
  - EDM-package (EDM), 12
  - Embed, 13, 14, 14, 17, 33
  - EmbedDimension, 13, 15, 15, 28
  - EvergladesFlow, 16

  - Lorenz5D, 16

  - make_block, 18, 18
  - make_surrogate_data, 19
  - MakeBlock, 17, 17
  - Multiview, 13, 19, 19
  - multiview, 21, 21, 22

  - paramecium_didinium, 23
  - PredictInterval, 13, 24, 24
  - PredictNonlinear, 13, 25, 25

  - s_map, 4, 28, 30, 31
  - s_map (simplex), 28
  - sardine_anchovy_sst, 26
  - Simplex, 13, 15, 24, 27, 27, 28, 32
  - simplex, 4, 22, 28, 28, 29, 30
  - SMap, 13, 25, 31, 31, 32, 33
  - SurrogateData, 33

  - tde_gp (block_gp), 3
  - TentMap, 34
  - TentMapNoise, 35
  - test_nonlinearity (block_gp), 3
  - Thrips, 35