

# Package ‘rEDM’

December 19, 2021

**Type** Package

**Title** Empirical Dynamic Modeling (EDM)

**Version** 1.10.0

**Date** 2021-12-19

**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](https://doi.org/10.1038/344734a0)>, the 'S-map' algorithm from Sugihara (1994) <[doi:10.1098/rsta.1994.0106](https://doi.org/10.1098/rsta.1994.0106)>, convergent cross mapping described in Sugihara et al. (2012) <[doi:10.1126/science.1227079](https://doi.org/10.1126/science.1227079)>, and, 'multiview embedding' described in Ye & Sugihara (2016) <[doi:10.1126/science.aag0863](https://doi.org/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, ccp] (<<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** 2021-12-19 01:10:02 UTC

**R topics documented:**

block_3sp . . . . .	2
block_inlp . . . . .	3
CCM . . . . .	5
ccm . . . . .	7
circle . . . . .	9
ComputeError . . . . .	10
compute_stats . . . . .	11
EDM . . . . .	11
Embed . . . . .	12
EmbedDimension . . . . .	13
EvergladesFlow . . . . .	14
Lorenz5D . . . . .	15
MakeBlock . . . . .	16
make_block . . . . .	17
make_surrogate_data . . . . .	18
Multiview . . . . .	18
multiview . . . . .	20
paramecium_didinium . . . . .	22
PredictInterval . . . . .	23
PredictNonlinear . . . . .	24
sardine_anchovy_sst . . . . .	25
Simplex . . . . .	26
simplex . . . . .	27
SMap . . . . .	30
SurrogateData . . . . .	32
TentMap . . . . .	33
TentMapNoise . . . . .	34
Thrips . . . . .	34
<b>Index</b>	<b>35</b>

---

 block\_3sp

*Time series for a three-species coupled model.*


---

**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 $t-2$
```

---

block\_inlp

*Perform generalized forecasting using simplex projection or s-map*

---

**Description**

`block_inlp` 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**

```
block_inlp(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)
```

**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'

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.)
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:

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

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 "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

```

block <- block_3sp
block_inlp(block[,2:4])

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

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

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

block <- block_3sp
x_t_pred = block_inlp(block, method = "s-map", theta = 3, columns =
c("x_t", "y_t"), first_column_time = TRUE, stats_only = FALSE, save_smap_coefficients = TRUE)

```

## 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, parameterList = 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.
parameterList	logical to add list of invoked parameters.
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.

if `includeData = TRUE` a named list with the following data.frames data.frame `Combo_rho` columns:

<code>LibMeans</code>	CCM mean correlations for each library size
<code>CCM1_PredictStat</code>	Forward cross map prediction statistics
<code>CCM1_Predictions</code>	Forward cross map prediction values
<code>CCM2_PredictStat</code>	Reverse cross map prediction statistics
<code>CCM2_Predictions</code>	cross map prediction values

If `includeData = TRUE` and `parameterList = TRUE` a named list "parameters" is added.

**References**

Sugihara G., May R., Ye H., Hsieh C., Deyle E., Fogarty M., Munch S., 2012. Detecting Causality in Complex Ecosystems. *Science* 338:496-500.

**Examples**

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

---

 ccm

---

*Convergent cross mapping using simplex projection*


---

**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.

$norm = 2$  (only option currently available) uses the "L2 norm", Euclidean distance:

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



**Value**

If `stats_only = 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 `stats_only = FALSE`: a named list with the following items: settings:

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

CCM1\_PredictStat and CCM2\_PredictStat `data.frames` have columns:

N	prediction number
E	embedding dimension
nn	number of nearest neighbors
tau	embedding time delay offset
LibSize	library size
rho	correlation coefficient
RMSE	root mean square error
MAE	maximum absolute error
lib	column name of the library vector
target	column name of the target vector

**Examples**

```
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)
```

**Description**

Time series of of 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:

rho	Pearson correlation
MAE	mean absolute error
RMSE	root mean square error

**Examples**

```
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	<i>Compute performance metrics for predictions</i>
---------------	--

---

### 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

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

---

EDM	<i>Empirical dynamic modeling</i>
-----	-----------------------------------

---

### 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. Functionality 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 measurement error in time series. *Nature*, 344:734-741.

Sugihara G. 1994. Nonlinear forecasting for the classification of natural time series. *Philosophical Transactions: Physical Sciences and Engineering*, 348 (1688) : 477-495.

Dixon, P. A., M. Milicich, and G. Sugihara, 1999. Episodic fluctuations in larval supply. *Science* 283:1528-1530.

Sugihara G., May R., Ye H., Hsieh C., Deyle E., Fogarty M., Munch S., 2012. Detecting Causality in Complex Ecosystems. *Science* 338:496-500.

Ye H., and G. Sugihara, 2016. Information leverage in interconnected ecosystems: Overcoming the curse of dimensionality. *Science* 353:922-925.

---

Embed

*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	<i>Optimal embedding dimension</i>
----------------	------------------------------------

---

**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.

**References**

Lorenz, Edward (1996). Predictability - A problem partly solved, Seminar on Predictability, Vol. I, ECMWF.

---

`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

<code>dataFrame</code>	input data.frame. The first column must be a time index or time values. The columns must be named.
<code>E</code>	embedding dimension.
<code>tau</code>	integer time delay embedding lag specified as number of time column rows.
<code>columns</code>	string of whitespace separated column name(s) in the input data to be embedded.
<code>deletePartial</code>	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	<i>Make a time delay offset block</i>
------------	---------------------------------------

---

### 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

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

### Arguments

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

```
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,
    lib = "", pred = "", D = 0, E = 1, Tp = 1, knn = 0,
    tau = -1, columns = "", target = "", multiview = 0, exclusionRadius = 0,
    trainLib = TRUE, excludeTarget = FALSE, parameterList = 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.
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.
parameterList	logical to add list of invoked parameters.
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

If parameterList = TRUE a named list "parameters" is added.

**References**

Ye H., and G. Sugihara, 2016. Information leverage in interconnected ecosystems: Overcoming the curse of dimensionality. *Science* 353:922-925.

**Examples**

```
data(block_3sp)
L = Multiview( dataFrame = block_3sp, lib = "1 100", pred = "101 190",
E = 2, columns = "x_t y_t z_t", target = "x_t" )
```

---

multiview

*Perform forecasting using multiview embedding*

---

**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**

```
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 calculations?)
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 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)
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.

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.

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:

col_i,...	col_j	column indices of the embedding
name_i,...	nam_j	column names of the embedding
rho		correlation of the projection
MAE		maximum absolute error of the projection
RMSE		root mean square error of the projection

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

### Examples

```
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	<i>Forecast interval accuracy</i>
-----------------	-----------------------------------

---

### Description

`PredictInterval` uses `Simplex` to evaluate prediction accuracy as a function of forecast interval `Tp`.

### Usage

```
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

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

### Value

A data.frame with columns `Tp`, `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.
numThreads	number of parallel threads for computation.
showPlot	logical to plot results.

### Details

The localisation parameter  $\theta$  weights nearest neighbors according to  $\exp(-\theta D / D_{\text{avg}})$  where  $D$  is the distance between the observation vector and neighbor,  $D_{\text{avg}}$  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)
```

---

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

*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,
        const_pred = FALSE, verbose = FALSE, validLib = vector(),
        generateSteps = 0, parameterList = FALSE, showPlot = FALSE)
```

**Arguments**

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

const_pred	logical to add a <i>constant predictor</i> 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.
generateSteps	number of predictive feedback generative steps.
parameterList	logical to add list of invoked parameters.
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 `EmbedDimension` can be applied. Accuracy statistics can be estimated by `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.

If `parameterList = TRUE`, a named list with "predictions" holding the data.frame, "parameters" with a named list of invoked parameters.

### References

Sugihara G. and May R. 1990. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. *Nature*, 344:734-741.

### Examples

```
data( block_3sp )
smplx <- Simplex( dataFrame = block_3sp, lib = "1 100", pred = "101 190",
  E = 3, columns = "x_t", target = "x_t" )
ComputeError( smplx $ Predictions, smplx $ Observations )
```

---

simplex

*Perform univariate forecasting*

---

### Description

`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.

`s_map` is similar to `simplex`, but uses the S-map algorithm to make forecasts.

**Usage**

```
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)

s_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**

<code>time_series</code>	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)
<code>lib</code>	a 2-column matrix, data.frame, 2-element vector or string of row indice pairs, where each pair specifies the first and last <i>*rows*</i> of the time series to create the library. If not specified, all available rows are used
<code>pred</code>	(same format as <code>lib</code> ), but specifying the sections of the time series to forecast. If not specified, set equal to <code>lib</code>
<code>norm</code>	the distance measure to use. see 'Details'
<code>E</code>	the embedding dimensions to use for time delay embedding
<code>tau</code>	the time-delay offset to use for time delay embedding
<code>tp</code>	the prediction horizon (how far ahead to forecast)
<code>num_neighbors</code>	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.)
<code>stats_only</code>	specify whether to output just the forecast statistics or the raw predictions for each run
<code>exclusion_radius</code>	excludes vectors from the search space of nearest neighbors if their <i>*time index*</i> is within <code>exclusion_radius</code> (NULL turns this option off)
<code>epsilon</code>	Deprecated.
<code>silent</code>	prevents warning messages from being printed to the R console
<code>theta</code>	the nonlinear tuning parameter (theta is only relevant if <code>method == "s-map"</code> )
<code>save_smap_coefficients</code>	specifies whether to include the <code>s_map</code> 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.

`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:

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

`model_output` named list with data.frames for each model. Columns include the time index, observations, predictions, and es

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:

<code>smap_coefficients</code>	data.frame with columns for the s-map coefficients
<code>smap_coefficient_covariances</code>	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 es

### Examples

```
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

*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

```
SMap(pathIn = "./", dataFile = "", dataframe = NULL,
     lib = "", pred = "", E = 0, Tp = 1, knn = 0, tau = -1,
     theta = 0, exclusionRadius = 0, columns = "", target = "", smapFile = "",
     embedded = FALSE, const_pred = FALSE, verbose = FALSE,
     validLib = vector(), generateSteps = 0, parameterList = FALSE,
     showPlot = FALSE)
```

### Arguments

<code>pathIn</code>	path to dataFile.
<code>dataFile</code>	.csv format data file name. The first column must be a time index or time values. The first row must be column names.
<code>dataFrame</code>	input data.frame. The first column must be a time index or time values. The columns must be named.

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.
embedded	logical specifying if the input data are embedded.
const_pred	logical to add a <i>constant predictor</i> 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.
generateSteps	number of predictive feedback generative steps.
parameterList	logical to add list of invoked parameters.
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.

If `parameterList = TRUE` a named list "parameters" is added.

**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**

Sugihara G. 1994. Nonlinear forecasting for the classification of natural time series. *Philosophical Transactions: Physical Sciences and Engineering*, 348 (1688):477-495.

**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", "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	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	<i>Time series for a tent map with <math>\mu = 2</math>.</i>
---------	--

---

**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	<i>Time series of tent map plus noise.</i>
--------------	--

---

**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	<i>Apple-blossom Thrips time series</i>
--------	---

---

**Description**

Seasonal outbreaks of *Thrips imaginis*.

**References**

Davidson and Andrewartha, Annual trends in a natural population of *Thrips imaginis* *Thysanoptera*, *Journal of Animal Ecology*, 17, 193-199, 1948.

# Index

## \* datasets

block\_3sp, [2](#)  
circle, [9](#)  
EvergladesFlow, [14](#)  
Lorenz5D, [15](#)  
paramecium\_didinium, [22](#)  
sardine\_anchovy\_sst, [25](#)  
TentMap, [33](#)  
TentMapNoise, [34](#)

## \* package

EDM, [11](#)

block\_3sp, [2](#)  
block\_lnlp, [3, 3](#)

CCM, [5, 5, 6, 7, 12](#)  
ccm, [7, 7, 8](#)  
circle, [9](#)  
compute\_stats, [11](#)  
ComputeError, [10, 10, 12, 27](#)

EDM, [11](#)  
EDM-package (EDM), [11](#)  
Embed, [12, 12, 16, 32](#)  
EmbedDimension, [12, 13, 13, 27](#)  
EvergladesFlow, [14](#)

Lorenz5D, [15](#)

make\_block, [17, 17](#)  
make\_surrogate\_data, [18](#)  
MakeBlock, [16, 16](#)  
Multiview, [12, 18, 18](#)  
multiview, [20, 20, 21](#)

paramecium\_didinium, [22](#)  
PredictInterval, [12, 23, 23](#)  
PredictNonlinear, [12, 24, 24](#)

s\_map, [3, 27, 29, 30](#)  
s\_map (simplex), [27](#)

sardine\_anchovy\_sst, [25](#)  
Simplex, [12, 13, 23, 26, 26, 27, 31](#)  
simplex, [3, 21, 27, 27, 28, 29](#)  
SMap, [12, 24, 30, 30, 31, 32](#)  
SurrogateData, [32](#)

TentMap, [33](#)  
TentMapNoise, [34](#)  
Thrips, [34](#)