NLDSPRED Software Manual

R.B. Toonen

Contents

/

Note: this manual is an excerpt of the appendix of my PhD thesis.

Chapter I

Software manual

This manual describes software that was developed to carry out analyses of time-series data obtained from nonlinear dynamical systems. The software can be obtained as an R package from https://github.com/toonenrb/nldspredr. The methods that are used to carry out the nonlinear analyses are all based upon calculating the predictive strength of functions operating on embeddings that were generated from these time series. Currently, the software provides functionality to carry out predictions based on combining values from nearest neighbors of a target point in an embedding, such as: Sequentially weighted global linear maps (SMAP) (Sugihara, 1994) and Convergent Cross Mapping (CCM) (Sugihara et al., 2012). It combines these prediction methods with several other techniques, such as: bundle embeddings to analyze data from synchronized systems (Stark, 1999), dewdrop embeddings to combine data from several similar nonlinear dynamical systems (Hsieh et al., 2008), and KD-trees to optimize the search for nearest neighbors in low-dimensional embeddings (Moore, 1991).

This manual consists of four parts. The first part provides an overview of embedding-based nonlinear dynamical systems methods and terminology that is used in

this manual. The second part describes the functionality of a dedicated C library called NLDSPRED (Nonlinear Dynamical Systems Prediction) that was developed to carry out embedding-based predictions. The third part contains an overview of R functions that were written as a frontend for the C library. Finally, part four contains an overview of data that can be retrieved from the output that is generated by NLDSPRED.

I.1 Embedding-based prediction

I.1.1 Background

When time series are available of each variable of a dynamical system, the combined values of each variable at a single point in time represent the state of the system at that time. For nonlinear dynamical systems, according to Takens' theorem (Takens, 1981), the state of the system can also be obtained by examining the historical values of a single variable of the system. The complete dynamical path of the system can accordingly be reconstructed by creating a set of lagged vectors from the time series. Such a path is commonly called an embedding. It is constructed as follows. Consider time-series data of a variable x consisting of T measurements, $x = (x_1, x_2, \ldots, x_T)$. To construct an embedding with dimension e, a total amount of $T - (e-1) \cdot d$ vectors can be obtained by taking $X_t = (x_t, x_{t-d}, x_{t-2d}, \ldots, x_{t-(e-1)d})$ at each value of t between $t = e \cdot d - 1$ and $t = T$, with d the distance, or lag size, between the values (Kantz & Schreiber, 2004). Here the convention is used that capital letters denote the point in the embedding and lowercase letters denote values from the time series. See Figure I.1. The parameters e and d are usually not known beforehand and need

to be determined first. A large variety of techniques have been established to do so, see Kantz and Schreiber (2004). Once a set of parameter values has been chosen, the corresponding embeddings can be constructed and used for further analysis.

I.1.2 Prediction

Many of the applicable analysis techniques use embedding-based predictions, where predicted values are compared with the real, observed, values. In general, these techniques work as follows. The embedding represents a path through time. Therefore, the point that is represented by the vector X_{t+1} can be regarded as the future of the point represented by the vector X_t. A property of many nonlinear dynamical embeddings is that neighbors of a point X_t – the target point, or predictee point – show a similar path through the embedding space as the path through the target point. That is, if $X_{t,1}, X_{t,2}, \ldots, X_{t,k}$ are k direct neighbors of X_t then the next points on the paths through these neighbors will still be close to X_t's future point, X_{t+1}. See Figure I.2. Therefore, it is possible to use these neighbors or their future values to obtain values for a variable that is associated with X_t, or X_t's future value. The techniques differ in terms of which associated variable is predicted, which variables are used in the embedding, and how the neighbors are used to predict that value. Some possibilities are described below.

A. Which associated values are used for prediction?

The associated value of an embedding vector is the value of a time-series variable that is stored together with the vector but is not part of that vector´s coordi-

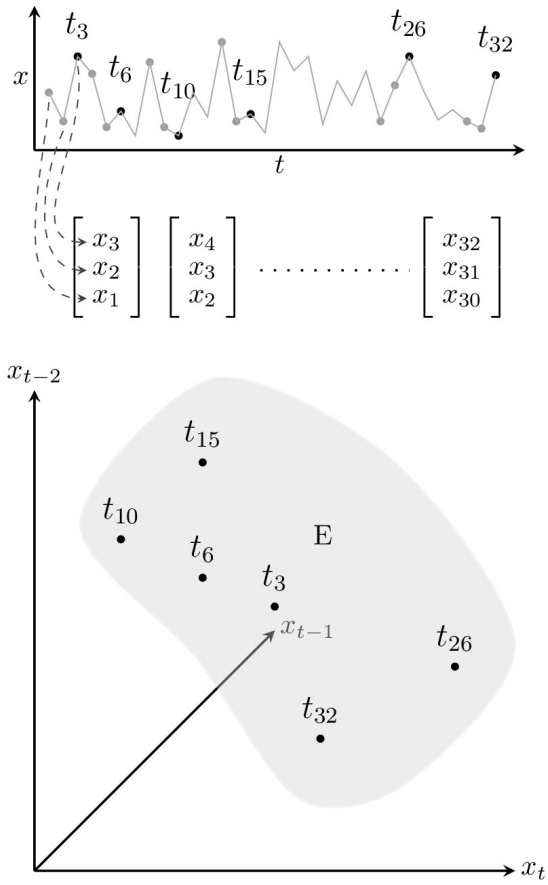

Figure I.1: Embedding construction.

Construction of an embedding. In this example, with e = 3 and d= 1, three-dimensional coordinate vectors are produced from time series x(t). The scalar components of each vector are obtained by taking lagged values of x at time t, t-1 and t-2. The embedding E is the set of all generated coordinate vectors. A dynamical path, connecting subsequent vectors in the embedding, has been omitted for the sake of clarity.

nates in the embedding. Each vector in the embedding has a unique associated value, not shared with other vectors. The options for the associated value are

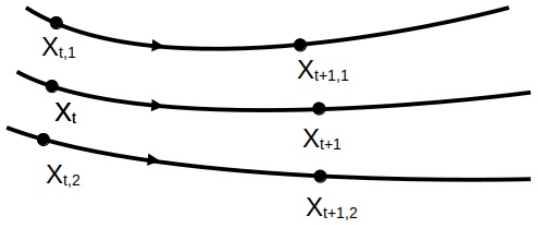

Figure I.2: Predicting future values.
Shown are a target point X_t and two of its closest neighbors, $X_{t,1}$ and $X_{t,2}$, along with their respective trajectories. Because these trajectories remain almost similar over a short period of time, the future value of X_t's neighbors, $X_{t+1,1}$ and $X_{t+1,2}$ remain quite close to X_t's future value, X_{t+1}.

described below.

A1. Use a future value of x itself

A future value x_{t+f} of x_t is predicted by considering the future values of the neighbors of X_t. Each of the neighbors $X_{t,1}, X_{t,2}, \ldots, X_{t,k}$ has its own corresponding time index, $T1, T2, \ldots, Tk$. Then, their future values in the time series, $x_{T1+f}, x_{T2+f}, \ldots, x_{Tk+f}$, may be used to predict the future value x_{t+f} , for example by taking averages with weights based on the distances between the neighbors and X_t.

A2. Use a future or contemporaneous value of another variable, y

This is very similar to predicting future values of x itself, but instead of associating a value x_{t+f} with each vector X_t, the technique associates a value y_{t+f} with each vector X_t. Here, y_{t+f} is a time-series value from another observed variable. The future value lag f can also be set to zero. In that case X's neighbors are used to predict the contemporaneous value of variable y. If such predictions yield positive results, this may be an indication that variables x and y are causally

related (Sugihara et al., 2012).

B. Which variables are used in the embedding?

B1. Univariate embeddings

In univariate embeddings, time-series values of only one observed variable are used to construct the coordinate vectors.

B2. Multivariate embeddings

The vectors in a multivariate embedding contain values from time series of two or more variables. For example, if time series have been obtained for three variables, x, y and z, then a 6-dimensional multivariate embedding could be constructed by taking vectors, $V_t = (x_t, y_t, x_{t-1}, y_{t-2}, z_{t-3}, z_{t-4})$. Of course, many other combinations are possible.

C. How are neighbors used to predict values?

C1. Use a weighted average of associated values

A prediction for a value that is associated with the target vector is obtained by calculating a weighted average of the associated values of the targets neighbors. Combined with option A1, the associated variable is taken from the time series that is also used to construct the embedding and therefore, the value that is predicted is the future value within that time series. It is also possible to use time series of another observed variable to obtain associated values (option A2). For example, if the embedding was constructed from variable x, the target vector Xt could be associated with a value y_{t+1} from a time series of variable y. The associated values of the neighbors, $X_{T1}, X_{T2}, \ldots, X_{Tk}$ would then be $y_{T1+1}, y_{T2+1}, \ldots, y_{Tk+1}$. These values would be averaged with weights depend-

ing on the distance between the neighbors and the target in the embedding.

C2. Use the neighbors to obtain parameters for a function that predicts the associated value

In this case, a function is used to predict the associated value from the coordinate values of the vector: $y = f(X_t)$. The parameters for function f are computed from the target's neighbors and their associated values.

I.1.3 SMAP and CCM

Several important nonlinear analysis techniques have been developed by Sugihara et al. (Hsieh et al., 2008; Sugihara, 1994; Sugihara et al., 2012) that consist of different combinations of the possibilities described above. For example, SMAP uses technique C2. More specifically, it uses a local linear model, resembling vector autoregression (VAR), with parameters obtained from the other vectors in the embedding. It can be combined with both A1 and A2 and both B1 and B2. In addition, when determining the values for the parameters of the prediction function f, the influence of each neighbor depends on its distance to the target according to a Gaussian weight function. The Gaussian's width is controlled by a parameter θ. Furthermore, SMAP uses all neighbors, which is: all embedding vectors except the target itself. CCM uses technique C1, in combination with A2 and B1. With CCM, analyses are carried out at gradually increasing library sizes. The library size is the number of vectors that are used for finding nearest neighbors of the target.

I.1.4 Additional techniques

In addition, the following techniques can be used on top of SMAP or CCM:

Dewdrop embeddings

In dewdrop embeddings, embedding vectors from different systems (or subjects) are combined (Hsieh et al., 2008). This can be done under the assumption that these systems (or subjects) have similar dynamics.

Bundle embeddings

Bundle embeddings may be used when a system contains a strong coupling with another system, that results in synchronization (Stark, 1999). This may lead to periodical trends in the target system's time series, that are not directly a result of the system's internal dynamics. The embedding based prediction techniques are then predicting the externally forced trend instead combined with variations due to internal dynamics. If the variance of the periodical trend is higher than the variance due to the internal dynamics, this may lead to false estimates for size of the prediction error. To analyze those time series, it may be necessary to group embedding vectors based on the value of an associated variable from the other system (e.g., time of day). Each such group of embedding vectors is called a bundle and can be regarded as a separate embedding.

I.2 The NLDSPRED software

I.2.1 Overview

The NLDSPRED software is a C library that provides researchers with several nonparametric analysis techniques based on the nonlinear dynamical systems theory that was described in I.1. When running, NLDSPRED writes values of almost every relevant parameter and computed variable to a binary log file. It also provides a framework to which a programmer can fairly easily add additional prediction functions. Although NLDSPRED can directly compute statistical measures associated with the prediction functions, its main purpose is to provide extensive output, which can be further analyzed in any other statistical software package (e.g., R packages). However, a dedicated frontend was developed in the R language, which has been included in an R package (nldspredr) that also contains the core functionality. All examples in this manual are based on this R package.

I.2.2 Time-series input and embedding creation

The main input for NLDSPRED are univariate or multivariate time series. Usually, an analysis is carried out using data from only one system or person at a time. In the case of multiple systems or persons, the results from individual analyses can be further analyzed afterwards to obtain information about groups of systems (or persons). An exception to this rule is when the assumption is made that the individual systems are comparable. In that case, dewdrop embeddings (see section 1.4.1 above) can be created by combining time-series data of multiple systems or persons.

Embeddings will be created based on user-specified embedding parameters. By using a simple syntax it is possible to specify a range of embeddings for NLDSPRED to create and analyze. Additionally, the user can specify a bundling variable to split the embedding into bundle embeddings (see section I.1.4 above).

I.2.3 Embedding structure

For each embedding specification, the program creates the corresponding embedding vectors from the time-series data, which are stored in memory. For each vector, the observed value of the associated predictee variable will also be stored in memory, so that it can be compared with the computed value afterwards. When bundle embeddings are used, the associated values of the bundle variables will also be stored. Sometimes, the user may want to know the values of time-series variables that are not used by the prediction functions, but are however required for further calculations. For example, if measurements have been made at different time of day (TOD), but TOD is not included in the embedding vectors, or used for bundle embeddings, it may still be that for further statistical analyses it is required that the TOD for each vector is known. This can be attained by specifying 'additional time-series variables' in the embedding definitions. NLDSPRED will associate the values of these variable with each embedding vector in memory, but will not use the variable in its prediction function. By writing the values of the additional variable(s) to the log file, NLDSPRED makes it possible to use them in subsequent analyses.

In NLDSPRED, vector coordinates, time values associated with the coordinates, predictee values, time values associated with the predictee values, bundle variable values, additional variable's values and time values associated with the additional variable's values are all stored separately as matrices and arrays (See Table I.1). The embedding structure is a collection of these matrices and an array with vector numbers that are used to uniquely identify each vector. This vector number is used in the log file to refer to individual vectors. Besides the above listed matrices and arrays, a separate data structure is stored for each embedding point individually. This point data structure consists of pointers into the data matrices of the embedding (See Table I.2). A pointer can be regarded as a reference to the row with the values for the vector associated with the point. For the remainder of this manual, the term 'vector' refers to one set of scalar values, specifying a location in a space, and the term 'point' refers to the data structure that is used in the program, containing references to several related vectors and other data.

I.2.4 Validation pairs

To compute the predictive strength of a prediction function, it is necessary to compare predicted values with observed values. This is done by selecting a library set and a predictee set, both containing references to points from the embedding. Vectors associated with points from the library set are used to predict values that are associated with points from the predictee set. A combination of one library set and one predictee set is called a validation pair. The user can choose from several different validation methods (see below) and – for some validation methods – can specify how many vectors each set should contain.

I.2.5 Predictor functions

NLDSPRED offers two prediction functions. Each function comes with its own parameters. The values for these parameters have to be specified a priori by the user. It is often the case that the user wants to find an optimal value for these parameters before applying the functions in further analyses. The software facilitates this by providing the option to supply a range and a step size for some of the parameters. Instead of using a fixed set of parameters, the program will perform the calculations using each possible combination of parameter values at the pre-specified incremental steps within the supplied ranges. Predictions can be made with two different functions: (1) exponentially weighted nearest neighbors (EWNN); and (2) total least squares estimated local linear vector autoregression (TLS-VAR) model.

Table I.1: Embedding structure

Data structure	Description
Vector number	Array with point/vector numbers (unique number for each point, used in output file): $(vn_1 \quad vn_2 \quad \dots \quad vn_n)$ n = number of vectors
Coordinates	Matrix with scalar values of each coordinate vector: $\begin{matrix} x_{1,1} & x_{1,2} & \dots & x_{1,e} \\ x_{2,1} & x_{2,2} & \dots & x_{2,e} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,e} \end{matrix}$ e = embedding dimension
Observed predictee values	Matrix with observed predictee values associated with each point: $\begin{matrix} y_{1,1} & y_{1,2} & \dots & y_{1,np} \\ y_{2,1} & y_{2,2} & \dots & y_{2,np} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,1} & y_{n,2} & \dots & y_{n,np} \end{matrix}$ np = number of predictee values per point. It is possible to have more predictee values when using the TLS estimation (explained below). The VAR parameters will then be based on a TLS estimation based on all predictee values. Usually however, np wil be 1.
Time values	Matrix with time values for each vector coordinate: $\begin{matrix} tx_{1,1} & tx_{1,2} & \dots & tx_{1,e} \\ tx_{2,1} & tx_{2,2} & \dots & tx_{2,e} \\ \vdots & \vdots & \ddots & \vdots \\ tx_{n,1} & tx_{n,2} & \dots & tx_{n,e} \end{matrix}$

Continued on next page

Table I.1 – continued from previous page

Data structure	Description
Time values of predictees	Matrix with time values associated with observed predictee values: $\begin{matrix} ty_{1,1} & ty_{1,2} & \dots & ty_{1,np} \\ ty_{2,1} & ty_{2,2} & \dots & ty_{2,np,..} \\ \vdots & \vdots & \ddots & \vdots \\ ty_{n,1} & ty_{n,2} & \dots & ty_{n,np} \end{matrix}$
Bundle values	Matrix with values of bundle variables associated with each point: $\begin{matrix} b_{1,1} & b_{1,2} & \dots & b_{1,nb} \\ b_{2,1} & b_{2,2} & \dots & b_{2,nb} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,nb} \end{matrix}$ nb = number of bundle values per point.
Additional values	Matrix with values of additional variables associated with each point: $\begin{matrix} a_{1,1} & a_{1,2} & \dots & a_{1,na} \\ a_{2,1} & a_{2,2} & \dots & a_{2,na} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,na} \end{matrix}$ na = number of additional values per point.
Additional time values	Matrix with time values for each additional variable's values: $\begin{matrix} ta_{1,1} & ta_{1,2} & \dots & ta_{1,na} \\ ta_{2,1} & ta_{2,2} & \dots & ta_{2,na} \\ \vdots & \vdots & \ddots & \vdots \\ ta_{n,1} & ta_{n,2} & \dots & ta_{n,na} \end{matrix}$

I.2.6 Exponentially weighted nearest neighbors

The EWNN function finds the nearest neighbors of the predictee vector. The function searches the library set for the nearest neighbors of the predictee vector that is selected from the predictee set. The function uses the associated predictee values of the nearest neighbors that were found to predict the associated value of the predictee vector. Each nearest neighbor predictee value is weighted according to the distance between the nearest neighbor vector and the predictee vector, using the function: weight = exp[-k * distance / reference_distance]. The number of nearest neighbors that is used by the function can be pre-specified by the user. It has been suggested that a good value to use is the dimension of the embedding plus one (Sugihara & May, 1990).

Table I.2: Embedding point (*)

Data structure	Description
Vector coordinates	The coordinates of point with index i: $(x_{i,1}, x_{i,2}, \ldots, x_{i,e})$ e = embedding dimension
Coordinate time values	Time value of each coordinate: $(tx_{i,1}, tx_{i,2}, \ldots, tx_{i,e})$
Predictee values	The values to predict, associated with point i: $(y_{i,1}, y_{i,2}, \ldots, y_{i,np})$ np = number of predictee values per point.
Predictee time values	Time value of each predictee value: $(ty_{i,1}, ty_{i,2}, \ldots, ty_{i,np})$
Bundle values	Values of bundle variables: $(b_{i,1}, b_{i,2}, \ldots, b_{i,nb})$ nb = number of bundle values per point.
Additional values	Values of additional variables: $(a_{i,1}, a_{i,2}, \ldots, a_{i,na})$ na = number of additional values per point.
Additional time values	Time values of additional variables: $(ta_{i,1}, ta_{i,2}, \ldots, ta_{i,na})$

(*) The elements of this structure actually consist of pointers into the rows in the matrices and the elements in the arrays of the embedding structure (Table I.1) that correspond to the embedding point.

I.2.7 TLS-VAR model

The TLS-VAR function uses a linear model to predict the predictee's value from the predictee vector's coordinate values. The parameters of the linear model are computed by using all vectors and associated predictee values from the library set, weighted according to the exponent of their distance to the predictee vector (REF). This results in an overdetermined system, which is solved by applying a total least squares (TLS) procedure (REF). The extent to which neighbors are used when computing the VAR parameters is determined by the variable θ. With $\theta = 0.0$, all library vectors have an equal influence in the VAR estimate. For linear systems, this should result in the best prediction accuracy. For nonlinear systems, it may be that the system shows local linear behavior. In that case, in the VAR parameter estimation, the nearby vectors should be assigned a higher weight than more distant vectors to obtain the best predictions. Higher values of θ correspond to the estimates becoming more local. Usually, θ ranges from 0.0 to values around 1.0 or higher. In a validation procedure, analyses can be rerun with different values of θ to find the value at which the best prediction accuracy is obtained (more global vs more local). This can be automated by letting the software go through a range of pre-specified θ values.

I.2.8 Main program flow

For each (bundle) embedding, NLDSPRED cycles through all values that were specified for the predictor function's parameter range. At each subsequent value, NLDSPRED cycles through all sets of validation pairs and applies the predictor function to each vector of the predictee set. See Algorithm I.1.

Algorithm I.1: Main program flow

```
foreach (bundle) embedding in embedding set do
    Write embedding parameters and vectors to log file;
    foreach parameter value in function parameter range do
        Write parameter value to log file;
        foreach library/predictee validation pair in set of validation pairs do
            Write vector IDs of all vectors in library and predictee set to
              log file;
            foreach predictee in predictee set do
                Use prediction function on vectors in library set (usually
                  nearest neighbors of predictee's associated vector) to
                  predict predictee's value;
                Write IDs of vectors that were used in the function to log
                  file;
                Write computed value to log file;
            end
            Optionally, write prediction statistics of validation set to log
              file;
        end
    end
end
```

I.2.9 Embedding specification syntax

When the aim of an analysis is to find an optimal embedding – that is, an embedding for which the prediction function shows the best prediction performance – the predictor function has to be applied to a range of embeddings, each with a different set of pre-specified parameters (e.g., lag size, dimension). The user can specify the parameters of the embeddings by (1) providing a fixed set of embedding definitions (i.e. strings of embedding parameter values) or (2) specifying a range of values for the embedding definitions.

I.2.10 Specifying a fixed set of embedding definitions

The user supplies a string with one or more embedding definitions. Each definition consists of coordinate variable definitions and predictor variable definitions. For each vector in the embedding, the vector coordinates are constructed by using the coordinate definitions. The values to be predicted are constructed from the predictor variable definitions.

The following characters are used: '|' separates embedding definitions, ':' separates the coordinate definitions group, the predictor definitions group and the additional variable definitions group from each other, ';' separates different variables within one group, and ',' separates variable names from lags and lags from each other. The group before the first ':' is used for the coordinate definitions. The group following the first ':' is used for the predictor definitions. When a second ':' has been specified, the group following it is used for the additional-variable definitions. Lag times are defined relative to the 'current time' of the vector – that is the time of the vector's foremost point in the time series. A lag time of 0 is regarded as the 'current time' for the vector. Positive lag times are times into the past. Times are not specified in the unit of time that is used, but in the number of time-series points.

Example

To construct an embedding of 3-dimensional lagged coordinate vectors from variable glucose with lags of 0, 2 and 5, and to predict the (future) value of glucose at lag -1, while also providing the value of additional variable TOD at lag 0 in the binary or textual data output, and to construct an embedding of 4-dimensional lagged coordinate vectors from variable glucose with lags of 1, 2

and 5 and of fat at lag 0, and to predict the (future) value of energy at lag -1 and the (current) value at lag 0, the following syntax can be used:

```
"glucose,0,2,5:glucose,-1:tod,0|glucose,1,2,5;fat,0:energy,0,-1"
 <-----------> <--------> <---> <-----------------> <--------->
 Coordinates   Prediction Addit  Coordinates        Prediction
 <----------------------------> <------------------------------>
 First embedding                Second embedding
```

I.2.11 Specifying a range of embedding definitions

When specifying a range of embedding parameter values, the following characters are used: '|' separates range definitions, ':' separates the groups of coordinate definitions, predictor definitions and additional-variable definitions from each other, ';' separates different variables within one group and ',' separates the variable name from the start lag, the start lag from the end lag, the end lag from the step size, and the step size from the method. Method can have a string value of 'trail' or 'shift'. In the trail method, a lag from the pre-specified list of values is added in each subsequent embedding (resulting in multiple embeddings with increasing numbers of lags). In the shift method, the program shifts to the next provided lag-value for each subsequent embedding (resulting in multiple embeddings with single but different lags).

Example The string "hr,3,7,2,trail" would result in subsequent construction of an embedding with lag 3 for hr, an embedding with lags 3 and 5 for hr, and an embedding with lags 3, 5 and 7 for hr. The string "hr,3,7,2,shift" would result in one embedding with lag 3 for hr, one embedding with lag 5 for hr

and one embedding with lag 7 for hr. When multiple groups have been defined, embeddings will be created for all combinations. For example, the string ”hr,2,3,1,shift;bp,4,6,2,trail” will create an embedding with lag 2 for hr and lag 4 for bp, an embedding with lag 2 for hr and lag 4 and 6 for bp, an embedding with lag 3 for hr and lag 4 for bp, and an embedding with lag 3 for hr and lag 4 and 6 for bp.

I.2.12 Vector exclusion

Sometimes it is necessary to exclude points from the library set, depending on the predictee vector that is being processed by the prediction function. The reason is that unwanted correlations could result from vectors sharing some of their coordinate values with the predictee, or from vectors being close in time to the predictee. NLDSPRED offers several methods to exclude points from the library. Applying an exclusion method leads to a decrease in the number of points that remain available in the library set. A choice for an exclusion method therefore depends on the size of the time series and the dimension of the embedding, as higher dimensions result in less points in the library set. Currently, the following exclusion methods are available.

Exclude vectors that share one or more coordinates with the predictee vector

Each coordinate in the predictee vector is taken from a point in the time-series data. When a vector from the library set has one or more of its coordinates taken from the same points as the predictee vector, this may lead to unwanted correlations. Therefore, there is the option to exclude these vectors from the library set.

Exclude vectors within a time window around the predictee vector

Using this option excludes vectors within a time window around the predictee vector. The size of the window can be specified using a separate parameter.

Exclude the predictee vector itself

With some validation methods, the predictee vector may also occur in the library set. By selecting this option, the predictee vector can be excluded from the library set when this is desired.

I.2.13 Validation set selection

NLDSPRED provides several methods to create library and predictee validation sets.

Leave-one-out cross-validation

When leave-one-out cross-validation (LOOC) is used, all embedding points are used in the predictee set as well as in the library set. When a prediction is computed for a point in the predictee set (the target, or predictee), that particular point is excluded from the library set.

K-fold cross-validation

The set of embedding points is split into k subsets. Each subset is used once as a predictee set. The other subsets are then combined into a library set. This can be combined with a Monte Carlo approach, where after a round of k predictee sets have been evaluated, the order of the points is reshuffled and the procedure is

repeated again.

Bootstrap validation

When bootstrap validation is used, the library set is repeatedly sampled from the embedding points. The number of bootstrap library sets can be specified as well as the number of points per bootstrap library set. The predictee set can be set equal to the library set (i.e. they contain the same points) or equal to the complete set of embedding vectors.

Convergent library validation

This way of selecting validation sets has been specifically developed with Sugihara's CCM method in mind (REF). In successive cycles, the size of the library is increased up to a maximum value. For each size, multiple library sets are generated from the complete set of embedding points. Two methods are available for this: (1) shift the library set through the complete set; or (2) choose points randomly for each shift. Using the first method for a library size N, for the first set the first N points are selected from the embedding. For the second set, points 2 to N+1 are selected, and so on. Using the second method, each time the N points are sampled randomly from the complete embedding.

I.2.14 Log file

NLDSPRED exports the values of many parameters and calculated variables to a non-human-readable binary log file. The layout of this file is compiler dependent. Subsequent processing of the file should therefore be done on the same type of computer, and using an NLDSPRED package that has been compiled

with the same compiler as was used to compile the NLDSPRED package that generated the binary file. The user specifies which categories of data NLDSPRED has to export. Data are written in the form of fixed size records, with each record containing several variables. The record types are specified in Table I.3. Each record starts with its record type. In the file this will be a number, in Table I.3, it is represented by a text code.

Table I.3: Record types in a NLDSPRED log file

Record type	Description
BP	Used to specify the bundle. Contains the bundle number.
BV	Used to specify the bundle value(s) that correspond to the bundle. Usually only one value, but there is the possibility to have bundles based on the values of several variables. Then, for each variable a separate BV record will be exported.
KP	Used to export the type of prediction function that is used.
KPEXP	Used to export the parameters of the exponentially weighted nearest neighbor (EWNN) prediction function.
KPTLS	Used to export the parameters of the Total least squares estimated local linear vector autoregression (TLS-VAR) prediction function.
NN	When the EWNN prediction function is used, for each target vector that is being processed, the program will export the vector numbers of all the nearest neighbors that are used in the calculation, together with the squared distance to the target.
PD	Used to export the predicted values for each target.
TG1	Used to export the vector number of the current target when nearest neighbors are exported.

Continued on next page

Table I.3 – continued from previous page

Record type	Description
TG2	Used to export the vector number of the current target when predicted values are exported.
VPM	When using the TLS-VAR prediction function, the option exists to center the set of library vectors around the origin (0, 0,..,0). In that case, the true center of the library vectors will have to be extracted from the predictee vector's coordinates before applying the VAR function that is used to predict the corresponding value. The VPM records are used to export the coordinate means that were used.
VPT	Used to export the VAR parameters that were used to predict the corresponding value of a target vector.
EMBPAR1	Used to export general embedding parameters.
EMBPAR2	Used to export the lag and prediction variable parameters.
EMBPAR3	Used to export range definitions, in case ranges are used to specify the embeddings.
VID	Used to export vector identifications, in case embeddings are constructed from time-series data of several subjects.
VAL	Used to export the time value and the value for each vector coordinate.
SPCL	When the convergent library validation is used, SPCL contains parameters associated with the validation method.
SPKF	When k-fold validation is used, SPKF contains parameters associated with the validation method.
SPLO	When LOOC validation is used, SPLO contains parameters associated with the validation method.
SPBT	When bootstrap validation is used, SPBT contains parameters associated with the validation method.

Continued on next page

Table I.3 – continued from previous page

Record type	Description
SH	Header for set parameters. Used to report parameters of the current set that is being processed. Used for the library as well as for the prediction set.
SD	Details of the current set. Used to identify each vector in the set. Each vector has an index number that may be different from the vector number in the embedding. The SD record is used to join these two numbers.
STNP	Contains the number of variables to predict per predictee vector. Usually 1.
STAT	Used to report some general prediction statistics per processed set.
ADHD	Header for additional value parameters. Contains the number of additional values that are associated with each vector.
ADDT	Details of additional values. For each additional value that is associated with a vector, an ADDT record will be exported.

The aim is to have minimal redundancy in the log file. To achieve this, each record type only contains variables that are not already reported in record types that are higher up in the program flow. For example, when a new embedding is constructed, NLDSPRED will first export an EMBPAR1 record that contains the embedding number. Then for each vector, it will first export the VID record, which contains the vector number but not the embedding number, and then it will export VAL records for each coordinate. The VAL records contain the coordinate indexes and values, but not the vector number. Data in commonly

used (wide-format) tables are usually organized differently. In this example, a table that would contain all vector-coordinate values for each embedding and each vector would contain the following columns: embedding number, vector number, vector coordinate, and coordinate value. To summarize, the contents of the NLDSPRED log file would be:

```
EMBPAR1,<embedding number of embedding 1>,<other variables>
VID,<vector number 1>,<other variables>
VAL,1,<value of scalar coordinate value 1>,<other variables>
VAL,2,<value of scalar coordinate value 2>,<other variables>
...
VAL,<dimension (e)>,<value of scalar coordinate value e>,<other variables>
VID,<vector number 2>,<other variables>
VAL,1,<value of scalar coordinate value 1>,<other variables>
VAL,2,<value of scalar coordinate value 2>,<other variables>
...
VAL,<dimension (e)>,<value of scalar coordinate value e>,<other variables>
[Repeated up to the number of vectors in the first embedding]
EMBPAR1,<embedding number of embedding 2>,<other variables>
[VID and VAL records for embedding 2]
...
```

Whereas a table in wide format would have the following columns:
`embedding number, vector number, val 1, val 2, ..., val e`
Where val 1 up to val e are the scalar coordinate values. The table would contain a row for each scalar value for each vector for each embedding, leading to considerable redundancy.

Since further analyses on the data will usually be carried out with software that uses table-like data structures (e.g., R or SPSS), an R function (NldsPredGetTable) is available to extract data in a table format. It functions as a 'variable collector' and collects values of variables that the user wants to export to the table. The user has to specify which variables are wanted in the table and at which record type a new table row is started. As long as the new line record has not been encountered in the log file, the function will keep reading new records from it and will store the values of the specified variables in memory. When a new value for the same specified variable is encountered, it overwrites the current value in memory. When the new line record is read, the function will create a table row with the current values of all the specified variables, including the values of specified variables in the new line record. To create a table for the example above, the function would have to create a new row after each VAL record that is read from the log file, and it would have to export the following variables: embpar1_emb_num, vid_vec_num, val_idx, val_val. The variables val_idx and val_val contain the coordinate index and the coordinate value, respectively. The first part of each variable name is equal to the name of the record type that contains it. See the specifications at the end of this manual.

I.2.15 Subsequent analyses

NLDSPRED assumes that the provided time-series data belong to one system or person. This implies that – in the case of group studies – the program has to be called for each participant separately. Furthermore, the output of NLDSPRED contains data for all the embeddings from the range of embeddings that have been created. Identifying the most optimal embedding and starting ad-

ditional NLDSPRED analyses based on that embedding is therefore a task of an additional layer of functionality. This task may be complicated by memory constraints of the computing system. For example, some of the analyses that were carried out using the software for applied research (REFS), resulted in approximately 1 GB of output data per participant. It was therefore difficult to completely adhere to the R philosophy of matrix and dataframe-oriented processing. Instead, the analyses were initially carried out on a per participant basis, using a loop. This included calling NLDSPRED, importing its output, selecting the optimal embedding, performing additional analyses based on these optimal embeddings, and – most importantly – carrying out data reduction by computing summary statistics. In additional steps, the size-reduced data of all participants were combined into data frames for further processing, in line with the R philosophy. This resulted in the program flow depicted in Algorithms I.2, I.3 and I.4.

Algorithm I.2: Typical analysis: compute predictions

Step I: Compute predictions;

foreach *participant* **do**

Call NLDSPRED to perform analyses on a range of embeddings;

Import the output data of NLDSPRED;

Perform additional analyses. For example:

-Find optimal embedding;

-Compute WC statistic for θ (= data reduction);

-Compute CIs for estimated model parameters (= data reduction);

-Determine optimal embedding;

Call NLDSPRED to perform additional analyses on the optimal embedding (for example, multivariate analyses);

Again perform the additional analyses, this may now also include analyses that compare results between different embeddings (e.g., the multivariate and the univariate one);

Store the results as dataframes in one result file for one participant;

end

Algorithm I.3: Typical analysis: combine participants

Step II: Combine data of all participants;

foreach *participant* **do**

    Read the participant's result file from step I;

    Add the dataframes to a list containing the dataframes for all participants;

end

Convert the list of dataframes to one overall dataframe, containing the results for all participants;

Write the overall dataframe to a file;

Algorithm I.4: Typical analysis: further processing

Step III: Further processing;

Read the file with the overall dataframe;

Use standard R functionality to create graphs and tables containing the results for all participants;

An important argument for choosing this order of processing is that it is easy to redo the analyses for only a few of the participants, in case something went wrong. This is convenient because it may take several hours per participant to process the results, depending upon the size of the embeddings, the number of values used for θ, and the number of bootstrap instances. The loop in step I can easily be adapted to carry out analyses only for specific participants. The already existing files for other participants will not be erased. Step II and III are much less time consuming than step I, so redoing these steps for all participants again will not be a problem.

I.3 R functions

`NldsPredFnExpOpts`	Set function options for exponential weighted nearest neighbor function

Description

Validate the parameter values and create a list containing parameters for the exponential function that uses the nearest neighbors to predict values. This list will be used as an input argument for `NldsPredRun`.

Usage

```
NldsPredFnExpOpts(expK, nnnAdd, excl, varWin, fnDenom)
```

Arguments

`expK`	The constant value that is used in the exponential function.

`nnnAdd`	To enclose a point in an n-dimensional space, at least n+1 nearest neighbors around it are needed. This parameter specifies how many additional nearest neighbors to add to the dimension number n. So, to use n+1 nearest neighbors, use a value of 1.
`excl`	Exclusion method. Options: none, timeCoord, timeWin, self. See exclusion section.
`varWin`	When the exclusion method is set to timeWin, use varWin to specify the size of the variable time window around the target. The unit of measurement is the number of time series points before or after the target.
`fnDenom`	The method that is used to calculate the denominator in the fraction in the exponent of the function. Options: min – the denominator is the distance between the closest nearest neighbor and the target; avgNn – the denominator is the average of the distances of the nearest neighbors to the target; avgLib – the denominator is the average distance between all points of the library set.

Value

This function returns a list of parameter settings.

Examples

```
# Create a list of parameters for the exponential
# prediction function.
fo <- NldsPredFnExpOpts (expK = 1.0, nnnAdd = 1,
          excl = "timeCoord", fnDenom = "min")
```

`NldsPredFnTlsOpts`	Set function options for the total least square vector autoregression (TLS-VAR) function

Description

Validate the parameter values and create a list containing parameters for the TLS-VAR function. This list will be used as an input argument for `NldsPredRun`.

Usage

```
NldsPredFnTlsOpts(thetaMin, thetaMax, deltaTheta, nnn,
excl, varWin, center, restrictPred, warnIsError, refMeth,
refXnn)
```

Arguments

`thetaMin`	For the TLS-VAR function, theta controls the amount of 'localness' of the VAR estimates With theta = 0.0, all library vectors have an equal influence in the VAR estimate. For linear systems, this should give the best results. For non-linear systems, it may be that the system shows local linear behavior In that case, in the VAR parameter estimation, the nearest vectors should be assigned a higher weight than more distant vectors. Higher values of theta correspond to the estimates becoming more local. Usually, theta can range from 0.0 to values around 1.0 or higher. In a validation procedure, different values of theta may be used to find the best results. It is possible to let the program go through a range of values for theta, ranging from thetaMin to thetaMax, with steps of deltaTheta.

`thetaMax`	See thetaMin.
`deltaTheta`	See thetaMin.
`nnn`	The TLS-VAR function normally uses all library vectors to compute the parameter values of the linear model. In the case of very large time series this may take a lot of time. Using this argument it is possible to use only a specified amount of nearest neighbor vectors.
`excl`	Exclusion method Options: none, timeCoord, timeWin, self. See exclusion section.
`varWin`	When the exclusion method is set to timeWin, use varWin to specify the size of the variable time window around the target. The unit of measurement is the number of time series points before or after the target.
`center`	Center embedding around the predictee vector. The coordinates of the predictee will all become 0. Options: TRUE, FALSE.
`restrictPred`	If this argument is greater than zero then predictions that are higher than this value will be discarded when calculating the statistics. This only pertains to the statistics that are calculated by NldsPred; not to additional statistics that are calculated afterwards by the user.
`warnIsError`	The TLS method may return warnings. If this argument is set to TRUE, the computed values will be regarded as erroneous and will be discarded. Options: TRUE, FALSE.
`refMeth`	Similarly to the exponential nearest neighbors function, TLS-VAR uses a reference distance in the denominator in an exponential weight function. Values can be: mean - to use the average distance between the predictee and all library vectors that are used in the TLS-VAR procedure; and xnn - to use the average distance between the predictee and xnn closest nearest neighbors.

`refXnn`	If refMeth has been set to xnn then refXnn should be set to the number of neighbors that have to be used to calculate the value of the denominator.

Value

This function returns a list of parameter settings.

Examples

```
# Create a list of parameters for the TLS-VAR
# prediction function.
fo <- NldsPredFnTlsOpts(thetaMin = 0.0, thetaMax = 2.0,
          deltaTheta = 0.1, excl = "timeCoord",
          center = TRUE, warnIsError = TRUE,
          refMeth = "mean")
```

`NldsPredGetTable`	Read the log file and return a data frame with the required fields.

Description

This function reads the log file that was generated by `NldsPredRun`. It returns a data frame with columns that were specified by the fieldNames argument. See the section I.4 for a detailed description.

Usage

```
NldsPredGetTable(logFileName, fieldNames, newRecname)
```

Arguments

`logFileName`	Name of the log file that was created by NldsPredRun.
`fieldNames`	String vector with names of fields to include in the data frame.
`newRecName`	Name of the lowest level record to include in one data frame row. After this record has been encountered in the log file and included in the current data-frame row, a new data-frame row will be created.

Value

This function returns a data frame object.

Examples

```
# Extract coordinates of points used in each embedding.
embpoints <- NldsPredGetTable (logFileName = fname,
                 fieldNames = c("embpar1_emb_label",
                                "embpar1_emb_num",
                                "embpar1_n_row",
                                "embpar1_e",
                                "embpar1_n_pre_val",
                                "embpar1_n_bundle_val",
                                "vid_vec_num",
                                "val_copr",
                                "val_idx",
                                "val_t",
```

```
                                        "val_val"),
                    newRecName = "VAL")
```

NldsPredRun	Starts the computations.

Description

This function starts the computations.

Usage

```
NldsPredRun(data, logFile, fnOpts, setOpts, time, id, bundle,
embLagDef, embRangeDef)
```

Arguments

data	Numeric matrix with time-series values. Each column corresponds to one variable. Time, id and bundle values are not included in the matrix. These are supplied through the time, id or bundle argument respectively. Column names should be set and correspond to the variable names that are used in the embLagDef or rangeDef arguments.
logFile	Name of the log file where output will be written to.
fnOpts	List of function options. Generated by NldsPredFnExpOpts or NldsPredFnTlsOpts.
setOpts	List of validation set options. Generated by NldsPredSetBootstrap, NldsPredSetConvergent, NldsPredSetKFold or NldsPredSetLooc.

`time`	When user-supplied time values are required in the log file, use this argument to supply a vector with integer values that can be transformed in real time values afterwards by the user. The length of this vector must be equal to the number of rows in the data matrix. Can be NULL.
`id`	When dewdrop embeddings are used, use this argument to supply the ID strings of each row in the data matrix. The length of this vector must be equal to the number of rows in the data matrix. Can be NULL. If it is not NULL, then the NldsPredRun will automatically use dewdrop embeddings.
`bundle`	When bundles are used, use this argument to supply the bundle values for each row in the data matrix. The number of rows in this numeric matrix must be equal to the number of rows in the data matrix. Can be NULL.
`embLagDef`	Embedding lag definition string. Can be NULL if embRangeDef is supplied.
`embRangeDef`	Embedding range definition string. Can be NULL if embLagDef is supplied.

Value

This function returns the name of the log file.

Examples

```
# Create a list of parameters for the exponential prediction
# function.
fo <- NldsPredFnExpOpts (expK = 1.0, nnnAdd = 1,
          excl = "timeCoord",
```

```
            fnDenom = "min")

# Create a list of validation set options for bootstrapping.
so <- NldsPredSetBootstrap (nBootstrap = 5000,
            libSizeIsEmbSize = TRUE)

# Matrix ds has two columns, x and y, with normalized
# time-series data. ti is a vector with integer times,
# corresponding to the rows in ds. The rows in ds, and the
# values in ti, must be ordered by increasing time value.
# Here, x is predicted from y, using several embedding
# dimensions (defined by embRangeDef).
fname <- NldsPredRun (data = ds, logFile = tempfile(),
              fnOpts = fo, setOpts = so, time = ti,
              embRangeDef = "y,0,3,1,trail:x,0")

# Extract coordinates of points used in each embedding.
embpoints <- NldsPredGetTable (logFileName = fname,
                  fieldNames = c("embpar1_emb_label",
                                 "embpar1_emb_num",
                                 "embpar1_n_row",
                                 "embpar1_e",
                                 "embpar1_n_pre_val",
                                 "embpar1_n_bundle_val",
                                 "vid_vec_num",
```

```
                                    "val_copr",
                                    "val_idx",
                                    "val_t",
                                    "val_val"),
                    newRecName = "VAL")

# Extract general statistics that were generated by
# NldsPredRun.
stats <- NldsPredGetTable(logFileName = fname,
             fieldNames = c("embpar1_emb_num",
                            "embpar1_e",
                            "spbt_boot_i",
                            "stat_n_pre_obs",
                            "stat_avg_obs",
                            "stat_var_pre",
                            "stat_var_obs",
                            "stat_cov_pre_obs",
                            "stat_mae_pre_obs",
                            "stat_rmse_pre_obs",
                            "stat_md_pre",
                            "stat_md_obs",
                            "stat_mdad_pre",
                            "stat_mdad_obs",
                            "stat_mdae_pre_obs"),
             newRecName = "STAT")
```

`NldsPredSetBootstrap`	Create a list of options for the bootstrap validation method.

Description

This function checks the arguments and creates a list of options for the bootstrap validation method. This list is used as an input argument for the `NldsPredRun` function.

Usage

```
NldsPredSetBootstrap(nBootstrap, libSize, preSetIsLib,
libSizeIsEmbSize, perAdditGroup)
```

Arguments

`nBootstrap`	Number of bootstrap sets to create.
`libSize`	Number of points to sample into each bootstrap set.
`preSetIsLib`	If this argument is set to TRUE, the prediction set will contain the same points as the library set. Options: TRUE, FALSE.
`libSizeIsEmbSize`	If this argument is set to TRUE, the number of points in the library set will always be set equal to the number of points in the embedding. If bundle embeddings are used, the sampled bootstrap set will contain as many vectors per bundle as the library set. Options: TRUE, FALSE.
`perAdditGroup`	When additional variables are used, and this argument is set to TRUE, the number of vectors with a specific value for the additional variable will be the same as in the library set. Options: TRUE, FALSE.

Value

This function returns a list of options.

Examples

```
so <- NldsPredSetBootstrap(nBootstrap = 2000,
          preSetIsLib = FALSE, libSize = 300)
so <- NldsPredSetBootstrap(nBootstrap = 1000,
          preSetIsLib = FALSE, libSizeIsEmbSize = TRUE)
```

`NldsPredSetConvergent`	Create a list of options for the convergent library validation method.

Description

This function checks the arguments and creates a list of options for the convergent library validation method that is used in Sugihara's CCM method. This list is used as an input argument for the `NldsPredRun` function.

Usage

```
NldsPredSetConvergent(libSizeMin, libSizeMax, libInc,
libIncFactor, shiftMethod, libShift, nBootstrap)
```

Arguments

`libSizeMin`	Minimum library size.
`libSizeMax`	Maximum library size.
`libInc`	Increment to use when increasing the library size.
`libIncFactor`	When the library size increases, it is often possible to use larger values for the increment that is used. If libIncFactor is greater than 0.0, libInc will be multiplied by this factor after each increment.
`shiftMethod`	The method that is used to create multiple library sets with the same size. Use option “shift” to move the selection window through the complete set of embedding points. If the right side of the window exceeds the embedding size, it will continue at the beginning. Use option ”random” to first put the embedding points in random order. Moving is done similar to the “shift” option. Use option ”bootstrap” to create library sets by sampling from the embedding.
`libShift`	If shiftMethod has been set to ”shift” or ”random”, then this argument specifies the number of points the selection window will move at each shift.
`nBootstrap`	The number of bootstrap sets to create when shiftMethod has been set to ”bootstrap”.

Value

This function returns a list of options.

Examples

```
so <- NldsPredSetConvergent(libSizeMin = 10,
          libSizeMax = 85, libInc = 1,
```

```
            libIncFactor = 1.1, shiftMethod = "random",
            libShift = 1)
so <- NldsPredSetConvergent(libSizeMin = 10,
            libSizeMax = 200, libInc = 1,
            libIncFactor = 1.3, shiftMethod = "bootstrap",
            nBootstrap = 3000)
```

`NldsPredSetKFold`	Create a list of options for the k-fold validation method.

Description

This function checks the arguments and creates a list of options for the k-fold validation method. This list is used as an input argument for the `NldsPredRun` function.

Usage

```
NldsPredSetKFold(kFold, nRep)
```

Arguments

`kFold`	The factor k for the k-fold validation method.
`nRep`	Number of repetitions. Before each repetition, the order of the points in the full set is reshuffled. Then a new cycle of k-fold validations is carried out.

Value

This function returns a list of options.

`NldsPredSetLooc`	Create a list of options for the LOOC validation method.

Description

This function creates a list of options for the LOOC validation method.

This list is used as an input argument for the `NldsPredRun` function.

Usage

```
NldsPredSetLooc()
```

Arguments

None

Value

This function returns a list of options.

I.4 Log file record specifications and hierarchy

I.4.1 Log file records

ADDT, additional values, details

`addt_idx`	Index of additional value.
`addt_addit_val`	Additional value.

ADHD, additional values, header

`adhd_n_addit_val`	Number of additional values.

BP, Bundle parameters

`bp_bundle_num`	Bundle number, within the current embedding.

BV, Bundle vector

`bv_vec_idx`	Index in bundle vector. Usually 0, but can be larger if multiple bundle variables are used.
`bv_vec_val`	Corresponding scalar bundle value.

EMBPAR1, Global embedding parameters

`embpar1_emb_num`	Embedding number.
`embpar1_emb_label`	Embedding label.
`embpar1_n_row`	Number of points in complete embedding.
`embpar1_e`	Dimension of embedding.
`embpar1_n_pre_val`	Number of variables to predict. Usually 1.
`embpar1_n_bundle_val`	Number of bundle variables. When bundles are used, usually 1.

EMBPAR2, Lag and prediction variable definitions

`embpar2_copr`	Coordinate variable definition (C), prediction variable definition (P), or bundle variable definition (B).
`embpar2_coord`	Coordinate index ([0,1,..,e-1]), predictee index, or bundle index
`embpar2_var_name`	Name of the variable associated with the coordinate, predictee or bundle.
`embpar2_lag`	Time lag associated with coordinate index, predictee or bundle.

EMBPAR3, Range definitions (when used)

`embpar3_seq`	Sequence number corresponding to position in range definitions.
`embpar3_dim`	Dimension corresponding to subrange.
`embpar3_tau`	Gap (tau) corresponding to subrange.

KP, Function type

`kp_fn_type`	Type of function [exp—tls].

KPEXP, Parameters of exponential function

kpexp_nnn_add	NNN added to embedding dimension.
kpexp_excl	Exclusion method.
kpexp_var_win	Exclusion window.
kpexp_denom_type	Denominator type in exponent of weighting function.
kpexp_exp_k	Value of constant in exponent of weighting function.

KPTLS, Parameters of TLS function

kptls_nnn	Number of nearest neighbors used (when not the complete library set).
kptls_excl	Exclusion method.
kptls_var_win	Exclusion window.
kptls_theta	Value of nonlinearity parameter theta.
kptls_center	Center augmented matrix before singular value decomposition.

NN, Nearest neighbors

nn_seq	NN number.
nn_num	Point number in embedding.
nn_sqdst	Squared distance to target.

PD, Predicted values

pd_pre_val_num	Prediction variable (usually 1).
pd_pre_val	Predicted value.
pd_obs_val	Observed value.

SD, Set details

sd_set_code	L=Library, P=Prediction
sd_vec_idx	Index of the point in the set.
sd_vec_num	Index of the point in the embedding. Corresponds to vid_vec_num.

SH, Set header

`sh_set_code`	L=Library, P=Prediction
`sh_set_num`	Set number (when there are multiple prediction / library sets for validation).
`sh_n_point`	Number of points in the set.

SPBT, Set parameters bootstrap

`spbt_boot_i`	Bootstrap number.

SPCL, Set parameters convergent library

`spcl_lib_size`	Library size.
`spcl_shift`	Shift value.
`spcl_boot_i`	Bootstrap number (when used).

SPKF, Set parameters k-fold

`spkf_k`	Value of k.
`spkf_repetition`	Repetition number (in case of repeated cross validation).

SPLO, Set parameters LOOC

`splo_dummy`	Dummy.

STAT, Statistics, per validation set/bundle

stat_pre_val_num	Sequence number of predicted variable. Usually 0, if only 1 variable is predicted.
stat_n_pre_obs	Number of prediction-observation pairs without missing values.
stat_avg_pre	Mean of predicted values.
stat_avg_obs	Mean of observed values.
stat_var_pre	Variance of predicted values.
stat_var_obs	Variance of observed values.
stat_cov_pre_obs	Covariance of predictions and observations.
stat_mae_pre_obs	Mean absolute error of predictions and observations.
stat_rmse_pre_obs	Root mean square error of predictions and observations.
stat_md_pre	Median of predicted values.
stat_md_obs	Median of observed values.
stat_mdad_pre	Median absolute deviation of predicted values to median.
stat_mdad_obs	Median absolute deviation of observed values to median.
stat_mdae_pre_obs	Median of absolute differences between predicted and observed values.

STNP, Number of variables to predict

stnp_n_pre_val	Number of variables to predict. Usually 1.

TG1, Prediction target, for function log

tg1_target_num	Target number (corresponds to vid_vec_num)

TG2, Prediction target, for predicted values log

tg2_target_num	Target number (corresponds to vid_vec_num)

VAL, Vector values, of points in an embedding

`val_copr`	Coordinate values, prediction values (observed), or bundle values. (C, P, or B).
`val_idx`	Vector coordinate index (or variable number for prediction or bundle values).
`val_t`	Corresponding time value.
`val_val`	Value.

VID, Vector identification, of points in an embedding

`vid_vec_num`	Point number, unique.
`vid_id`	Person or system ID of point, in case of dewdrop embeddings.

VPM, Means (in TLS function), used to subtract when centering is used

`vpm_copr`	Coordinate values, prediction values (C, P).
`vpm_var_num`	Variable number.
`vpm_var_mean`	Mean value.

VPT, VAR parameter estimates (in TLS function)

`vpt_target_num`	Target number (corresponds to `vid_vec_num`).
`vpt_pre_val_num`	Prediction variable number.
`vpt_var_coord`	VAR vector coordinate index. 0 = constant offset, higher values are indexes of linear model parameters.
`vpt_var_val`	Computed value of VAR parameter.

I.4.2 Hierarchy

A run of NLDSPRED starts a loop in which, at each step, a new embedding is created. If bundles are used, then, for each embedding, a loop is started in which, at each step, a new bundle is processed. For each bundle or embedding, a loop is started in which, at each step, a new set of function parameters is generated, when a range of function parameters has been specified. For each set of function parameters, a loop is started in which, at each step, a new vali-

dation set is created. For each validation set, a loop is started in which, at each step, a new predictee is processed by applying the selected prediction function, which may also contain additional loops. The newRecName argument in the `NldsPredGetTable` function, must therefore be set to the name of the log file record of which fields are required in the log file and that is created in the innermost loop. That is, the log file record at the lowest level in the hierarchy of required log file records. The overall hierarchy, of all log file records is as follows.

```
KP
| EMBPAR1
| | EMBPAR2
| | EMBPAR3
| | VID
| | | VAL
| | BP
| | | BV
| | | KPEXP
| | | | [SPCL|SPKF|SPLO|SPBT]
| | | | | SH
| | | | | | SD
| | | | | TG1
| | | | | | NN
| | | | | TG2
| | | | | | PD
| | | | | STNP
| | | | | | STAT
| | | | | ADHD
| | | | | | ADDT
| | | | | | | STNP
| | | | | | | | STAT
| | | KPTLS |
| | | | [SPCL|SPKF|SPLO|SPBT]
| | | | | SH
| | | | | | SD
| | | | | TG1
| | | | | | NN
| | | | | VPM
| | | | | VPT
| | | | | TG2
```

```
| | | | | | PD
| | | | | STNP
| | | | | | STAT
| | | | | ADHD
| | | | | | ADDT
| | | | | | | STNP
| | | | | | | | STAT
```

When specifying fields for the `NldsPredGetTable` function, it is not possible to have fields of records that are at the same level go in one table. When that is required, generate two separate tables, by running `NldsPredGetTable` twice, and join the tables afterwards.

References

Hsieh, C. H., Anderson, C., & Sugihara, G. (2008). Extending nonlinear analysis to short ecological time series. *The American Naturalist*, *171*(1), 71–80. DOI: 10.1086/524202.

Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis* (Vol. 7). Cambridge university press. DOI: 10.1017/CBO9780511755798.

Moore, A. W. (1991). An intoductory tutorial on kd-trees.

Stark, J. (1999). Delay embeddings for forced systems. I. Deterministic forcing. *Journal of Nonlinear Science*, *9*(3), 255–332. DOI: 10.1007/s003329900072.

Sugihara, G. (1994). Nonlinear forecasting for the classification of natural time series. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, *348*(1688), 477–495. DOI: 10.1098/rsta.1994.0106.

Sugihara, G., May, R., Ye, H., Hsieh, C. H., Deyle, E., Fogarty, M., & Munch, S. (2012). Detecting causality in complex ecosystems. *Science*, *338*(6106), 496–500. DOI: 10.1126/science.1227079.

Sugihara, G., & May, R. M. (1990). Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. *Nature*, *344*(6268), 734–741. DOI: 10.1038/344734a0.

Takens, F. (1981). Detecting strange attractors in turbulence. In D. Rand & L.-S. Young (Eds.), *Dynamical systems and turbulence, warwick 1980* (pp. 366–381, Vol. 898). Springer Berlin Heidelberg. DOI: 10.1007/BFb0091924.