NlinTS: An R Package For Causality Detection in Time Series

by Youssef Hmamouche

Abstract The causality is an important concept that is widely studied in the literature, and has several applications, especially when modelling dependencies within complex data, such as multivariate time series. In this article, we present a theoretical description of methods from the **NlinTS** package, and we focus on causality measures. The package contains the classical Granger causality test. To handle non-linear time series, we propose an extension of this test using an artificial neural network. The package includes an implementation of the Transfer entropy, which is also considered as a non-linear causality measure based on information theory. For discrete variables, we use the classical Shannon Transfer entropy, while for continuous variables, we adopt the k-nearest neighbors approach to estimate it.

Introduction

The study of dependencies between variables is an important step in the analysis of multi-variate time series. Not surprisingly, it can be exploited in causal discovery for financial and neuroscience datasets, in feature selection to determine the most important variables as inputs of prediction models, *etc*. Standard measures like correlation and mutual information are very used for analyzing relationships between time series. Because these measures are symmetrical, they do not provide enough information concerning the transfer of information over time from one variable to another one. Therefore, in cases where we are interested in approximating non-symmetrical dependencies between variables, causality is more adequate than correlation measures.

In the literature, two main causality measures have been well investigated in the field of time series analysis; the Granger causality test (Granger, 1980), and the Transfer entropy (Schreiber, 2000). The Granger causality is based on the principle that a variable causes another variable if it contains useful information in terms of prediction. Consequently, it is mainly linked to the idea of using of a prediction model to test the causality. The Transfer entropy in the other hand is based on information theory and has gained an increasing attention during recent years. It measures the flow of information between variables using the conditional Shannon entropy. Although these two measures seem radically different, an interesting finding has been presented in Barnett et al. (2009) showing that they are equivalent for variables that follow a normal distribution. In addition, Transfer entropy is considered as a non-linear alternative for the Granger causality, since it does not model the relationships between variables using a statistical model, instead, it is based on information theory.

This article covers a theoretical description of methods implemented in the **NlinTS** package (Hmamouche, 2020). Particularly, we focus on methods and models that are related to causality measures. This package includes the Granger causality test. To deal with non-linear dependencies between time series, we propose an non-linear extension of the Granger causality test using feed-forward neural networks. The package includes also an implementation of Transfer entropy. Two versions are provided, one for discrete variables, and the second is an estimate for continuous variables based on the k-nearest neighbors approach (Kraskov et al., 2004). Therefore, We detail the Granger causality test, the proposed non-linear Granger causality test, the VARNN (Vector Auto-Regressive Neural Network) model, since it is used in the later. Then, we represent the Transfer entropy, including the original formulation and the continuous estimation, starting by the estimate of the entropy and the mutual information, because they will be useful to understand the Transfer entropy estimator.

It is worth to mention that there are several R packages that contain an implementation of the Granger causality test, such as **vars** (Pfaff, 2008), **lmtest** (Zeileis and Hothorn, 2002). However, for Transfer entropy, especially for the continuous estimation, we found only the **RTransferEntropy** package (Simon et al., 2019). The approach used for estimating the Transfer entropy for continuous variables is based on discretization methods, by transforming continuous variables to discrete, then, applying Shannon Transfert entropy. In this paper, our approach is based on the same principle proposed in Kraskov et al. (2004) to estimate the mutual information, which inherits from the Kozachenko-Leonenko estimator of the Shannon entropy.

The organization of the paper is as follows, the two first sections are for the theoretical formulation of the causality tests and the Transfer entropy measures. The third section provides R code examples of the presented measures, illustrating the usage of the implemented methods. Finally, the last section summarizes this paper.

The Granger causality test

The Granger causality test (Granger, 1980) is the classical method to test the causality between time series. To test if a variable X causes another variable Y, the principle of this test is to predict Y using its own history, and to predict it using it history plus the history of the variable X, and finally to evaluate the difference between these two situations to see if the added variable has some effect on the predictions of the target variable.

Formally, two VAR (p) (Vector Auto-Regressive) models are considered. The first one uses the precedent values of Y, and the second uses both passed values of X and Y in order to predict Y:

$$\text{Model}_1 \quad Y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i} + U_t, \tag{1}$$

$$\text{Model}_2 \quad Y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{i=1}^{p} \beta_i X_{t-i} + U_t, \tag{2}$$

where p is the lag parameter, $[\alpha_0, \ldots, \alpha_p]$ and $[\beta_0, \ldots, \beta_p]$ are the parameters of the models, and U is a white noise error term.

To quantify the causality, we have to evaluate the variances of the errors of Model_1 and Model_2. In this case, the Granger causality index (GCI) can be used, and it is expressed as follows:

$$\text{GCI} = log\left(\frac{\sigma_1^2}{\sigma_2^2}\right), \tag{3}$$

where σ_1^2 and ${\sigma_2}^2$ are the variances of the errors of Model_1 and Model_2 *resp*. In order to evaluate the statistical significance of the difference between these variances, the Fisher test can be used, where the statistic is as follows:

$$F = \frac{\left(RSS_1 - RSS_2\right)/p}{RSS_2/\left(n - 2p - 1\right)}.$$

RSS_1 and RSS_2 are the residual sum of squares related to Model_1 and Model_2 *resp.*, and n is the size of the lagged variables. Two hypotheses have to be considered:

- H_0: $\forall i \in \{1, \ldots, p\}, \beta_i = 0$,
- H_1: $\exists i \in \{1, \ldots, p\}, \beta_i \neq 0$.

H_0 is the hypothesis that X does not cause Y. Under H_0, F follows the Fisher distribution with $(p, n - 2p - 1)$ as degrees of freedom.

A non-linear Granger causality test

Using artificial neural networks (ANNs) may be very important when computing causalities, especially for time series that change non-linearly over time. We take advantage from the characteristics of ANNs and propose an implementation of an extended version of the Granger causality test using the VARNN model. Before describing the proposed causality test, let us first present briefly the VARNN model which is also available in the package as a prediction model.

The VARNN model: Consider a training dataset that consists of a multivariate time series containing one target variable Y, and k predictor variables $\{Y_1, \ldots, Y_k\}$. The VARNN (p) model is a multi-layer perceptron neural network model that takes into account the p previous values of the predictor variables and the target variable (Y) in order the predict future values of Y. We made this choice to allow for the possibility of predicting each target variable with a specific set of predictors, since target variables do not necessarily have the same predictors. First, the model reorganizes the data in a form of a supervised learning form with respect to the lag parameter. The optimization algorithm used to update the weights of the network is based on the Stochastic Gradient Descent (SGD) algorithm. The Adam algorithm can also be used to update the learning rate while using SGD (Kingma and Ba, 2015). The global function of the VARNN (p) can be written as follows:

$$Y_t = \Psi_{nn}\left(Y_{t-1}, \ldots, Y_{t-p}, \ldots, Y_{k(t-1)}, \ldots, Y_{k(t-p)}\right) + U_t, \tag{4}$$

where Ψ_{nn} is the network function, and U_t represents the error terms.

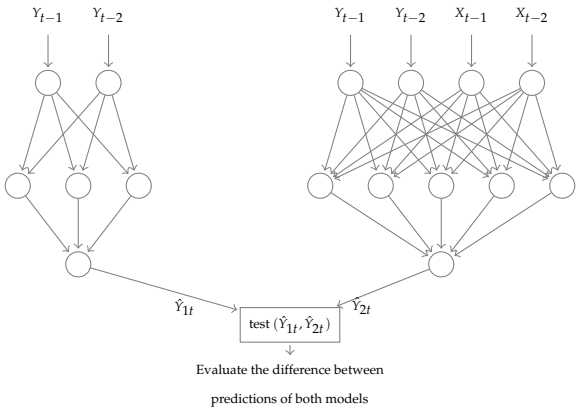

Figure 1: Illustration of the ANN model for the Granger causality test.

A causality test using the VARNN model: Consider two variables X and Y. Similarly to the Granger causality, to test the causality from X to Y, two prediction models are considered, the first takes into account the passed values of the target time series, and the second takes the passed values of the target and the predictor time series,

$$\text{Model}_1: \quad Y_t = \Psi_{1nn}\left(Y_{t-1}, \ldots, Y_{t-p}\right) + U_t, \tag{5}$$

$$\text{Model}_2: \quad Y_t = \Psi_{2nn}\left(Y_{t-1}, \ldots, Y_{t-p}, X_{t-1}, \ldots, X_{t-p}\right) + U_t, \tag{6}$$

where Ψ_{1nn} and Ψ_{2nn} are the network functions of Model_1 and Model_2 *resp.*, using the VARNN model. Then, we evaluate the difference between these two models by comparing the residual sum of squares of their errors, and the evaluation is carried out using the Fisher test to examine the null hypothesis (the hypothesis that X does not cause Y). Figure 1 shows an illustration of the used structure of the causality model.

The difference compared to the classical test, is that instead of using 2 VAR models (univariate and bivariate), two VARNN models are used. Therefore, we have to change the statistic of the Fisher test because there are more parameters in the VARNN models than in the VAR model. In this case, the statistic of test is as follows:

$$F = \frac{\left(RSS_1 - RSS_2\right) / \left(d_2 - d_1\right)}{RSS_2 / \left(n - d_2\right)},$$

where d_1 and d_2 are the number of parameters of the univariate and the bivariate model *resp*. They depend on the chosen structure (number of layers and of neurons).

Let us emphasize an important point about this causality. It is evident that computing causalities using ANNs may has the classical drawback of increasing the computational time. This is not exactly precise in some cases, because suppose that we have a large number of time series and we have to compute causalities between all variables. Also, suppose that relationships between variables change over time. Therefore, this implies that we need to recalculate the causalities periodically or after each change. In addition, the basic formulations of the classical causality measures (Granger causality test and Transfer entropy) are not adaptive, which means they do not make it possible to update the new values by using the old ones. In the other hand, with ANNs, the first computation of causalities may be slow compared to the Granger test or the Entropy Transfer, but if we have new observations in the time series, the model adapts more quickly thanks to the learning properties of ANNs.

Transfer entropy

Transfer entropy (Schreiber, 2000) between two time series X and Y, measures the information flow from X to Y. It was developed to overcome the main drawback of mutual information, which provides the common information between two variables (symmetric measure), but does not consider the transfer of information from one variable to the other. To avoid this problem, time delay parameters

are included in the equation of the mutual information to specify the direction of information:

$$\begin{aligned} T_{X\to Y} &= \sum_{Y_t, Y_t^q, X_t^p} P\left(Y_t, Y_t^q, X_t^p\right) log\left(\frac{P\left(Y_t \mid Y_t^q, X_t^p\right)}{P\left(Y_t \mid Y_t^q\right)}\right) \\ &= I\left(Y_t; X_t^p | Y_t^q\right), \end{aligned}$$

where $Z_t^l = (Z_{t-1}, \ldots, Z_{t-l})$ for $Z = X, Y$, p, q are the time delay parameters for X and Y *resp.*, P represents the probability, and I represents the mutual information symbol. The transfer entropy can also be seen as the difference between two conditional entropies, where in the first one, only past values of Y are used, and in the second, both X and Y are considered:

$$\begin{aligned} TE_{X\to Y} = & H\left(Y_t | \left(Y_{t-1}, \ldots, Y_{t-q}\right)\right) \\ & - H\left(Y_t | \left(Y_{t-1}, \ldots, Y_{t-q}\right), \left(X_{t-1}, \ldots, X_{t-p}\right)\right), \end{aligned}$$

where H represents the conditional entropy. Note that this expression resembles, in some sense, the principle of the Granger causality test which compares two prediction models.

A continuous estimation of Shannon Transfer entropy

In this section, we describe the estimation of Transfer entropy based on the k-nearest neighbors. First, we show the entropy estimator represented in Kraskov et al. (2004). Then, we show the mutual information estimator that is based on an extended formulation based on the same principal. Then, we use this approach to estimate the Transfer entropy.

Entropy estimation The basic approach for estimating the entropy of continuous variables is based on binning the data, in order to get back to the classical definition of Shannon entropy. However, more efficient approaches are proposed by estimating directly the continuous entropy:

$$H(X) = -\int p(x) log(x) dx,$$

where p represents the density function of X. One estimation of the continuous entropy of a random variable X with n realizations is the expected value of $log(p(X))$:

$$\hat{H}(X) = -\frac{1}{n}\sum_{1}^{n} log\left(\hat{p}(x_i)\right)$$

The main point of the Kozachenko-Leonenko estimator to approximate $log(p(x_i))$ by considering $p(x_i)$ constant in the sphere centered at x_i, with radius the distance from x_i to the k-nearest neighbors of each point. We do not show the details of the mathematical proof, but just the obtained formula:

$$\hat{H}(X) = \Gamma(n) - \Gamma(k) + log(c) + \frac{m}{n}\sum_{i=1}^{n} d_i, \tag{7}$$

where Γ is the gamma function, m is the dimension of X, *i.e*, the number of variables, d_i is twice the distance from x_i to its k^{th} neighbor, and c is the volume of the unit ball of dimension m. To compute the distances between two points x_i and x_j, we use the max norm, $|x_i - x_j|$, therefore, $c = 1$, and $log(c) = 0$. In the rest of the equations, for simplicity, we neglect this term.

Mutual Information estimation The mutual information between two variables X and Y having n observations can be expressed as follows:

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{8}$$

It is possible to adopt the Kozachenko-Leonenko approach to estimate the mutual information. In this case, we need to estimate the individual entropy of each variable and the joint entropy. For the joint entropy, it can be computed using the same way by considering the joint space spanned by X and Y. Let $z_i = (x_i, y_i)$ for $i \in [1, n]$, and d_i be the distance for z_i to its k^{th} neighbor. The estimate of the

joint entropy can be expressed as follows:

$$\hat{H}(X,Y) = \Gamma(n) - \Gamma(k) + \frac{m_x + m_y}{n} \sum_{i=1}^{n} d_i, \tag{9}$$

where m_x and m_y are the dimensions of X and Y.

In Kraskov et al. (2004), two new methods have been proposed to improve the Kozachenko-Leonenko estimator for mutual information. The first method is based on the idea that when estimating $H(X)$ and $H(Y)$, we do not have to use the same k as used in the joint entropy, but instead, it is more precise to use the number of neighbors of each variable separately. Thus, the estimate of the individual entropy, of X for example, is the following:

$$\hat{H}(X) = \Gamma(n) - \frac{1}{n} \sum_{i=1}^{n} \Gamma(nx(i) + 1) + \frac{m}{n} \sum_{i=1}^{n} d_i, \tag{10}$$

where $nx(i)$ is the number of points where the distance from X_i is strictly less than $d_i/2$. As for Y, $\hat{H}(Y)$ is computed with the same way. Finally, based on Equations 8, 9 and 10, the mutual information estimator is as follows:

$$\hat{I}(X;Y) = \Gamma(k) + \Gamma(n) - \frac{1}{n} \sum_{i=1}^{n} \left(\Gamma(n_x(i) + 1) + \Gamma\left(n_y(i) + 1\right)\right) \tag{11}$$

Following the same method and generalizing the previous formulation to l variables$\{X_1, \ldots, X_l\}$, the multivariate mutual information estimator is as follows:

$$\hat{I}(X_1, \ldots, X_l) = \Gamma(k) + (l-1)\Gamma(n) - \frac{1}{n} \sum_{i=1}^{n} (\Gamma(n_1(i) + 1) + \cdots + \Gamma(n_l(i) + 1)), \tag{12}$$

where $n_j(i)$, for $(j,i) \in [1,l] \times [1,n]$, is the number of points where the distance from the point X_{ji} is strictly less than $d_i/2$.

The motivation behind the second estimator of mutual information presented in Kraskov et al. (2004) is that the Kozachenko-Leonenko estimation of the joint entropy ($H(X,Y)$ in the bi-variate case) may be more precise than the first estimator if we consider that the density is constant in hyper-rectangles instead of hyper-cubes. Based on this remark, the second estimate of the mutual information of l variables $\{X_1, \ldots, X_l\}$, with n observations, can be expressed as follows:

$$\hat{I}(X_1, \ldots, X_l) = \Gamma(k) + \frac{l-1}{k} + (l-1)\Gamma(n) - \frac{1}{n} \sum_{i=1}^{n} (\Gamma(n_1(i) + \cdots + \Gamma(n_l(i)))), \tag{13}$$

where $n_j(i)$, for $(j,i) \in [1,l] \times [1,n]$, is the number of points where the distance from the X_{ji} is less (not strictly) than $d_{ji}/2$, and d_{ji} is the distance from X_{ji} to its k^{th} neighbor.

Transfert entropy estimation Let us use the first strategy used by Kraskov for mutual information estimation to estimate the Transfer entropy. Let X and Y be two time series. The goal is to estimate the Transfer entropy from X to Y, with time delay parameters p and q *resp*.

$$\hat{TE}_{X \to Y} = \hat{H}\left(Y_t | Y_{t-1}, \ldots, Y_{t-q}\right) - \hat{H}\left(Y_t | \left(Y_{t-1}, \ldots, Y_{t-p}\right), \left(X_{t-1}, \ldots, X_{t-p}\right)\right). \tag{14}$$

Consider the following notations :

- $Y_t^m = \{Y_{t-1}, \ldots, Y_{t-q}\}$
- $X_t^m = \{X_{t-1}, \ldots, X_{t-p}\}$
- $Y_t^f = \{Y_t, Y_t^m\}$
- $X_t^f = \{X_t, X_t^m\}$
- $Z_t^m = \{Y_t^m, X_t^m\}$
- $Z_t^f = \{Y_t^f, Y_t^m, X_t^m\}$

We can rewrite then Equation 14 as follows:

$$\begin{aligned} \hat{TE}_{X\to Y} &= \hat{H}\left(Y_t, Y_p\right) - \hat{H}\left(Y_t|X_p, Y_p\right), \\ &= \hat{H}\left(Y_t, Y_p\right) - \hat{H}\left(Y_p\right) - \hat{H}\left(Y_t, X_p, Y_p\right) + \hat{H}\left(X_p, Y_p\right) \\ &= \hat{H}\left(Y_t^f\right) - \hat{H}\left(Y_t^m\right) - \hat{H}\left(Z_t^f\right) + \hat{H}\left(Z_t^m\right). \end{aligned} \tag{15}$$

The maximum joint space is defined by $Z_t^f = \{Y_t, Y_t^m, X_t^m\}$. Consider that Z_t^f contains n observations. The first step is to computes the distances d_i, *i.e.*, the distance from the point z_i to its k^{th} neighbor, for $i \in [1, n]$. In the same way as estimating mutual information, we compute the maximal joint entropy $\hat{H}\left(Z_t^f\right)$ using the Kozachenko-Leonenko estimator, and the other terms by projecting the number of neighbors in each marginal space using the Kraskov approach:

$$\begin{aligned} \hat{H}\left(Z_t^f\right) &= -\Gamma(k) + \Gamma(n) + \frac{p+q+1}{n}\sum_{i=1}^{n} d_i, \\ \hat{H}\left(Y_t^f\right) &= -\frac{1}{n}\sum_{i=1}^{n}\Gamma\left(n_{y_f}(i)+1\right) + \Gamma(n) + \frac{p+1}{n}\sum_{i=1}^{n} d_i, \\ \hat{H}\left(Y_t^m\right) &= -\frac{1}{n}\sum_{i=1}^{n}\Gamma\left(n_{y_m}(i)+1\right) + \Gamma(n) + \frac{p}{n}\sum_{i=1}^{n} d_i, \\ \hat{H}\left(Z_t^m\right) &= -\frac{1}{n}\sum_{i=1}^{n}\Gamma\left(n_{z_m}(i)+1\right) + \Gamma(n) + \frac{p+q}{n}\sum_{i=1}^{n} d_i, \end{aligned}$$

where $n_{y_f}(i)$, $n_{y_m}(i)$ and $n_{z_m}(i)$ are the numbers of points where the distance from the point Y_i^f, Y_i^m, and Z_i^m *resp.*, is strictly less than $d_i/2$, for $i \in [1, n]$. By replacing each oh these terms in Equation 15, we obtain:

$$\begin{aligned} \hat{TE}_{X\to Y} = \Gamma(k) - \Gamma(n) - \frac{(p+1) - p - (p+q+1) + (p+q)}{n}\sum_{i=1}^{n} d_i \\ + \frac{1}{n}\sum_{i=1}^{n}\Big(-\Gamma\left(n_{y_f}(i)+1\right) + \Gamma\left(n_{y_m}(i)+1\right) - \Gamma\left(n_{z_m}(i)+1\right)\Big), \end{aligned} \tag{16}$$

By simplifying this expression, the Transfer entropy estimator can be expressed as follows:

$$\hat{TE}_{X\to Y} = \Gamma(k) - \Gamma(n) + \frac{1}{n}\sum_{i=1}^{n}\Big(\Gamma\left(n_{y_m}(i)+1\right) - \Gamma\left(n_{y_f}(i)+1\right) - \Gamma\left(n_{z_m}(i)+1\right)\Big). \tag{17}$$

And this is the classical Transfer entropy estimator investigated and discussed in Vicente et al. (2011); Lizier (2014); Zhu et al. (2015).

Normalizing the Transfer entropy

The values obtained by the Transfer entropy (TE) are not normalized, and practically, it is hard to quantify the causality in this case. Normalizing the values of TE between 0 and 1 simplifies the interpretation of the amounts of transferred information. For discrete data, The Transfer entropy from a variable X to a variable Y has a maximum value $H\left(Y_t|Y_t^m\right)$. Thus, the normalized TE (NTE) can be obtained by dividing TE by its maximum value:

$$NTE = \frac{\hat{H}\left(Y_t|Y_t^m\right) - \hat{H}\left(Y_t|Y_t^m, X_t^m\right)}{\hat{H}\left(Y_t|Y_t^m\right)} \tag{18}$$

In Gourévitch and Eggermont (2007), a preparation step is added to compute NTE to consider data that contain noise. It consists of subtracting first the average of TE by shuffling the variable X several times (rearranged it randomly):

$$NTE = \frac{TE_{X\to Y} - \sum_{i=1}^{n} TE_{X_{shuffled}\to Y}}{\hat{H}\left(Y_t|Y_t^m\right)}$$

In the package, we implemented just the first normalization (*cf.* Equation 18), because the second

one depends on the way of shuffling the variable X. But it can be obtained easily by computing the NTE with the original variables, and the average of NTE with several shuffled variables of X.

Concerning continuous Transfer entropy, the term $\hat{H}(Y_t|Y_t^m)$ may be negative, which means that if we apply the same method to normalize the discrete TE, we will not obtain values in $[0,1]$. To avoid this problem, we adopt another approach presented in Duan et al. (2013):

$$NTE = \frac{TE_{X\to Y} - \sum_{i=1}^{n} TE_{X_{shuffled}\to Y}}{H_0 - \hat{H}(Y_t|Y_t^m)},$$

where H_0 is the maximum entropy of Y by considering the uniform distribution, *i.e.*, $H_0 = log(Y_{max} - Y_{min})$, and Y_{max} and Y_{min} are the maximum and the minimum values of Y.

R code examples

In this section, we demonstrate worked examples about the usage of the methods implemented in the package and discussed theoretically in the two previous sections. We use financial time series from the package **timeSeries** (Wuertz et al., 2017). We will present the classical Granger causality test, the VARNN prediction model, and the proposed non-linear Granger causality test. These functionalities are provided via **Rcpp** modules. We present also the functions associated to Transfer entropy measures, including the discrete and continuous estimate. Since other entropy measures are implemented, we will present them as well, such as the entropy and the mutual information.

The Granger causality test

The `causality.test` module is based on an **Rcpp** module. The two first arguments of the constructor of this module are two numerical vectors, (the goal is to test if the second vector causes the first one). The third argument is the lag parameter, which is an integer value. The last argument is logical (false by default) for the option of making data stationary using the Augmented Dickey-Fuller test, before performing the causality test.

```
library (timeSeries)
library (NlinTS)
data = LPP2005REC
# Construct the causality model from the second column to the first one,
# with a lag equal to 2, and without taking into account stationarity
model = causality.test (data[,1], data[,2], 2, FALSE)
```

The `causality.test` module has a `summary` method to show all the results of the test, and 3 properties: the Granger causality index; `gci` (*cf.* 2.2), the statistic of the test (`Ftest`), and the p-value (the probability of non causality) of the test (`pvalue`).

```
# Compute the causality index, the Ftest, and the pvalue of the test
model$summary ()
model$gci
model$Ftest
model$pvalue
```

The VARNN model

The `varmlp` module represents the implementation of the VARNN model. It is an **Rcpp** module, where the constructor takes as arguments a numerical Dataframe. Each column represents a variable, and the first column is the target variable. Note that the Dataframe may contain one column. In this case, the model will be univariate (ARNN model). The second argument is the lag parameter, then, a numerical vector representing the size of the hidden layers of the network, then, an integer argument for the number of iterations to train the model. Other arguments with default values are available about using the bias neuron, the activation functions to use in each layer, the learning rate, and the optimization algorithm. More details about these arguments can be found in the manual of the package (Hmamouche, 2020).

```
library (timeSeries)
library (NlinTS)
```

```
# Load the data
data = LPP2005REC

# The lag parameter
lag = 1

# The training set
train_data = data[1:(nrow (data) - 1), ]

# Build and train the model
model = varmlp (train_data, 1, c(10,5), 100)
```

The `varmlp` module has 3 methods. The method named `forecast` compute predictions from an input `dataframe`, in other words, to test the model. And a method `train` update the parameters of the model from new data.

```
# Predict the last row of the data
predictions = model$forecast (train_data)

# Show the predictions
print (predictions[nrow (predictions),])

# Update the model (two observations are required at least since lag = 1)
model$train (data[nrow (data) - lag: nrow (data)])
```

The non-linear Granger causality test

Similarly to the previous test, the `nlin_causality.test` is an **Rcpp** module. The two first arguments of the constructor of this module are two numerical vectors, (the goal is to test if the second causes the first). The third argument is the lag parameter. The next two arguments are two numerical vectors representing the size of the hidden layers used in models 1 and 2, *resp*. The next argument is an integer for the number of the iterations to train the networks. Similarly to the `varmlp` model, other arguments with default values are available about the bias neuron, the activation functions, the learning rate, and the optimization algorithm. The manual of the package contain more details concerning these arguments (Hmamouche, 2020). The following is an example of using the non-linear causality test:

```
library (timeSeries)
library (NlinTS)
data = LPP2005REC
# Build and train the  model
model = nlin_causality.test (data[,1], data[,2], 2, c(2), c(4))
```

The `nlin_causality.test` module returns the same values as the `causality.test`; a `summary` method to show all the results of the test, and 3 properties; the Granger causality index (`gci`), the statistic of the test (`Ftest`), and the p-value of the test (`pvalue`).

```
# Compute the causality index, the Ftest, and the pvalue of the test
model$summary ()
model$gci
model$Ftest
model$pvalue
```

The discrete entropy

The function `entropy_disc` permits to compute the Shannon entropy, where the first argument is a discrete vector, and the second argument is the logarithm function to use (log_2 by default):

```
library (NlinTS)
# The entropy of an integer vector
print (entropy_disc (c(3,2,4,4,3)))
```

The continuous estimation of the entropy

The function `entropy_disc` permits to compute the continuous estimation of Shannon entropy, where the first argument is a numerical vector, and the second argument is the number of neighbors (see 2.4.1):

```
library (timeSeries)
library (NlinTS)
# Load data
data = LPP2005REC
# The entropy of the first column with k = 3
print (entropy_cont (data[,1], 3))
```

The discrete mutual information

The function `mi_disc` permits to compute the Shannon multivariate mutual information, where the first argument is an integer dataframe, and the second argument is the logarithm function to use (log_2 by default):

```
library (NlinTS)
# Construct an integer dataframe with 2 columns
df = data.frame (c(3,2,4,4,3), c(1,4,4,3,3))
# The mutual information between columns of df
mi = mi_disc (df)
print (mi)
```

The continuous estimation of the mutual information

The function `mi_cont` permits to compute the continuous estimate of the mutual information between two variables. The two first arguments are two vectors, and the third argument is the number of neighbors (see 2.4.1):

```
library (timeSeries)
library (NlinTS)
# Load data
data = LPP2005REC
# The mutual information between of the two first columns of the data with k = 3
print (mi_cont (data[,1], data[,2], 3))
```

The discrete Transfer entropy

The function associated to the discrete TE is named `te_disc`. The two first arguments are two integer vectors. Here we allow the two time series to have different lag parameters. Therefore, the second two arguments are the lag parameters associated to the first and the second arguments *resp*. The next argument indicates the logarithm function to use (log_2 by default). The last argument is logical for the option of normalizing the value of TE, with a false value by default. The `te_disc` function returns the value of Transfer entropy from the second variable to the first variable:

```
library (NlinTS)
# The transfer entropy between two integer vectors with lag = 1 to 1
te = te_disc (c(3,2,4,4,3), c(1,4,4,3,3), 1, 1)
print (te)
```

The continuous estimation of the Transfer entropy

The associated function is named `te_cont`. The two first arguments are two vectors. Then, the second two arguments are the associated lag parameters for the first and the second arguments *resp*. The fifth argument is the number of neighbors. The last argument is logical for the option of normalizing the value of TE, with a false value by default. The `te_cont` function returns the value of Transfer entropy from the second variable to the first one:

```
library (timeSeries)
library (NlinTS)
```

```
# Load data
data = LPP2005REC
# The transfer entropy between two columns with lag = 1 and k = 3
te = te_cont (data[,1], data[,2], 1, 1, 3)
print (te)
```

Conclusion

In this paper, we have presented methods of our **NlinTS** package for computing causalities in time series. We have considered two main measures well studied in the literature, the Granger causality test and the Transfer entropy. The Transfer entropy is originally formulated for discrete variables. For continuous variables, we adopted a k-nearest neighbors estimation based on the same strategy used to estimate the Mutual Information in Kraskov et al. (2004). To deal with non-linear time series, we have proposed another causality measure as an extension of the Granger causality test using an artificial neural network. Finally, we showed examples for the usage of these methods.

Bibliography

L. Barnett, A. B. Barrett, and A. K. Seth. Granger causality and transfer entropy are equivalent for gaussian variables. *Phys. Rev. Lett.*, 103:238701, Dec 2009. URL https://doi.org/10.1103/PhysRevLett.103.238701. [p1]

P. Duan, F. Yang, T. Chen, and S. L. Shah. Direct causality detection via the transfer entropy approach. *IEEE Transactions on Control Systems Technology*, 21(6):2052–2066, Nov 2013. ISSN 1063-6536. URL https://doi.org/10.1109/TCST.2012.2233476. [p7]

B. Gourévitch and J. J. Eggermont. Evaluating Information Transfer Between Auditory Cortical Neurons. *Journal of Neurophysiology*, 97(3):2533–2543, Mar. 2007. ISSN 0022-3077. URL https://doi.org/10.1152/jn.01106.2006. [p6]

C. W. J. Granger. Testing for causality. *Journal of Economic Dynamics and Control*, 2:329–352, Jan. 1980. ISSN 0165-1889. URL https://doi.org/10.1016/0165-1889(80)90069-X. [p1, 2]

Y. Hmamouche. *NlinTS: Models for Non Linear Causality Detection in Time Series*, 2020. URL https://CRAN.R-project.org/package=NlinTS. R package version 1.4.2. [p1, 7, 8]

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. [p2]

A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004. URL https://doi.org/10.1103/PhysRevE.69.066138. [p1, 4, 5, 10]

J. T. Lizier. Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI*, 1:11, 2014. ISSN 2296-9144. URL https://doi.org/10.3389/frobt.2014.00011. [p6]

B. Pfaff. Var, svar and svec models: Implementation within r package vars. *Journal of Statistical Software, Articles*, 27(4):1–32, 2008. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v027.i04. [p1]

T. Schreiber. Measuring Information Transfer. *Physical Review Letters*, 85(2):461–464, July 2000. URL https://doi.org/10.1103/PhysRevLett.85.461. [p1, 3]

B. Simon, D. Thomas, P. Franziska J., and Z. David J. Rtransferentropy — quantifying information flow between different time series using effective transfer entropy. *SoftwareX*, 10(100265):1–9, 2019. URL https://doi.org/10.1016/j.softx.2019.100265. [p1]

R. Vicente, M. Wibral, M. Lindner, and G. Pipa. Transfer entropy–a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1):45–67, Feb. 2011. ISSN 1573-6873. URL https://doi.org/10.1007/s10827-010-0262-3. [p6]

D. Wuertz, T. Setz, and Y. Chalabi. *timeSeries: Rmetrics - Financial Time Series Objects*, 2017. URL https://CRAN.R-project.org/package=timeSeries. R package version 3042.102. [p7]

A. Zeileis and T. Hothorn. Diagnostic checking in regression relationships. *R News*, 2(3):7–10, 2002. URL https://CRAN.R-project.org/doc/Rnews/. [p1]

J. Zhu, J.-J. Bellanger, H. Shu, and R. Le Bouquin Jeannès. Contribution to Transfer Entropy Estimation via the k-Nearest-Neighbors Approach. *Entropy*, 17(6):4173–4201, June 2015. URL https://doi.org/10.3390/e17064173. [p6]

Youssef Hmamouche
Aix Marseille Université, Université de Toulon, CNRS, LIS, UMR7020, Marseille, France
Aix Marseille Université, CNRS, LPL, UMR7309, Aix-en-Provence, France
youssef.hmamouche@lis-lab.fr