

# From Time Series to Networks in R with the **ts2net** Package

Leonardo N. Ferreira 

Max Planck Institute for Human Development

---

## Abstract

Network science established itself as a prominent tool for modeling time series and complex systems. This modeling process consists of transforming a set or a single time series into a network. Nodes may represent complete time series, segments, or single values, while links define associations or similarities between the represented parts. R is one of the main programming languages used in data science, statistics, and machine learning, with many packages available. However, no single package provides the necessary methods to transform time series into networks. This paper presents **ts2net**, an R package for modeling one or multiple time series into networks. The package provides time series distance functions that can be easily computed in parallel and in supercomputers to process larger data sets and methods to transform distance matrices into networks. **Ts2net** also provides methods to transform a single time series into a network, such as recurrence networks, visibility graphs, and transition networks. Together with other packages, **ts2net** permits the use of network science and graph mining tools to extract information from time series.

*Keywords:* time series, networks, graphs, network science, complex networks, R.

---

## 1. Introduction

Networks are one of the most prominent tools for modeling complex systems (Mitchell 2006). Complex systems are commonly represented by a set of time series with interdependencies or similarities (Silva and Zhao 2018). This set can be modeled as a network where nodes represent time series, and links connect pairs of associated ones. Associations are usually measured by time series distance functions, carefully chosen by the user to capture the desired similarities. Thus, the network construction process consists of two steps. First, the distance between all pairs of time series is calculated and stored in a distance matrix. Then, this distance matrix is transformed into an adjacency matrix. The network topology permits not just the analysis of the small parts (nodes) in the system but also their relationship (edges). This powerful model allows network science and graph mining tools to extract information from different domains such as climate science (Boers *et al.* 2019; Ferreira *et al.* 2021a), social science (Ferreira *et al.* 2021b), neuroscience (Bullmore and Sporns 2009), and finance (Tumminello *et al.* 2010).

In recent years, networks have also been applied to model a single time series. The goal is to transform one time series into a network and apply network science techniques to perform time series analysis (Silva *et al.* 2021). Different modeling approaches have been proposed, and the general idea is to map single values (Lacasa *et al.* 2008), time windows, transitions

Campanharo *et al.* (2011), or recurrence Donner *et al.* (2010) as nodes and their relationship by links. Then, the general goal is to use network science and graph mining tools to extract information from the modeled time series.

R is one of the most used programming languages in statistics, data science, machine learning, complex networks, and complex systems. Different packages provide distance functions and network analysis, but no single package provides the necessary tools to model one or multiple time series as networks. This paper presents **ts2net**, a package to construct networks from time series in R. **ts2net** permits the use of network science to study time series. Together with other R packages, **ts2net** permits distance calculations, network construction, visualization, statistical measures extraction, and pattern recognition using graph mining tools. Most of the key functions are implemented using multi-core programming, and some useful methods make **ts2net** easy to run in supercomputers and computer clusters. The package is open-access and released under the MIT license.

In the following sections, I present some background concepts related to time series network modeling (Sec. 2), the **ts2net** package (Sec. 3), and two applications using real data (Sec. 4), followed by some final remarks.

## 2. Background Concepts

In summary, the whole network construction process involves the calculations of time series distance functions, represented by a distance matrix, and the transformation of this matrix into a network. This section presents the distances and the network construction methods available in the **ts2net** package.

### 2.1. Distance functions

Time series comparison is a fundamental task in data analysis. This comparison can be estimated using similarity or distance functions (Deza and Deza 2009; Esling and Agon 2012; Wang *et al.* 2013). Some research domains use the concept of similarity, while others prefer distance functions. Similarity and distance are inverse concepts that have the same usefulness. Here in this paper, for simplicity, I use the distance concept. However, it is important to remember that a similarity function can be generally transformed into distance functions and vice-versa. The distance or metric  $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a function that for all time series  $X, Y, Z \in \mathcal{X}$ , the following properties hold:

- *non-negativity*:  $d(X, Y) \geq 0$ ,
- *symmetry*:  $d(X, Y) = d(Y, X)$ ,
- *reflexity*:  $d(X, Y) = 0 \iff X = Y$ ,
- *triangular difference*:  $d(X, Y) \leq d(X, Z) + d(Z, Y)$ .

Some of the time series comparison measures used in **ts2net** are not true metrics because they do not follow one or more required properties above (Deza and Deza 2009). However, for simplicity reasons, this paper uses the term “distance” to generally refer to any of these

functions. The rest of Sec. 2.1 reviews the main distance functions implemented in the **ts2net** package.

### *Correlation-based distances*

The Pearson Correlation Coefficient (PCC)  $r$  is one of the most used association measures (Asuero *et al.* 2006). Given two time series  $X$  and  $Y$  (both with length  $T$ ), and their respective mean values  $\bar{X}$  and  $\bar{Y}$ , the PCC is defined as

$$r(X, Y) = \frac{\sum_{i=1}^T (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^T (X_i - \bar{X})^2 (Y_i - \bar{Y})^2}}. \quad (1)$$

The PCC can be transformed into a distance function by considering the absolute value of  $r$  as follows:

$$d_{\text{cor\_abs}}(X, Y) = 1 - |r(X, Y)|. \quad (2)$$

In this case (Eq. 2), both strong correlations ( $r \approx 1$ ) and strong anti-correlation ( $r \approx -1$ ) lead to  $d_{\text{cor\_abs}}(X, Y) \approx 0$ . This distance is particularly useful if both correlated and anti-correlated time series should be linked in the network. Conversely, if only strong positive correlations or strong negative correlation should be mapped into links in the network, then Eq. 3 or Eq. 4 can be used instead.

$$d_{\text{cor\_pos}}(X, Y) = 1 - \max(0, r(X, Y)) \quad (3)$$

$$d_{\text{cor\_neg}}(X, Y) = 1 - \max(0, -r(X, Y)) \quad (4)$$

Cross-correlation is also a widely used measure to find lagged associations between time series. Instead of directly calculating the PCC, the cross-correlation function (CCF) applies different lags (displacement) to the series and then returns the correlation to each lag. Instead of returning a single correlation value, as in the PCC, the CCF returns an array of size  $2 \cdot \tau_{\text{max}} + 1$ , where  $\tau_{\text{max}} \in \mathbb{N}$  is the maximum lag defined by the user. Each element of this array corresponds to the  $r(X, Y_\tau)$  between the time series  $X$  and  $Y_\tau$ , which is the  $Y$  lagged by  $\tau \in [-\tau_{\text{max}}, \tau_{\text{max}}]$  units.

The CCF can be transformed into a distance function in the same way as the PCC (Eqs. 2, 3, and 4). The maximum absolute (Eq. 5), positive (Eq. 6), or negative (Eq. 7) correlations  $r(X, Y_\tau), \forall \tau \in [-\tau_{\text{max}}, \tau_{\text{max}}]$ .

$$d_{\text{ccor\_abs}}(X, Y) = 1 - \left( \max_{\tau \in [-\tau_{\text{max}}, \tau_{\text{max}}]} |r(X, Y_\tau)| \right). \quad (5)$$

$$d_{\text{ccor\_pos}}(X, Y) = 1 - \left( \max_{\tau \in [-\tau_{\text{max}}, \tau_{\text{max}}]} \max(0, r(X, Y_\tau)) \right) \quad (6)$$

$$d_{\text{ccor\_neg}}(X, Y) = 1 - \left( \max_{\tau \in [-\tau_{\text{max}}, \tau_{\text{max}}]} \max(0, -r(X, Y_\tau)) \right) \quad (7)$$

One crucial question is the correlation statistical significance. The significance test avoids spurious correlations and false links in the network. One way to test consists of using the Fisher z-transformation defined as (Asuero *et al.* 2006):

$$z(r) = \frac{1}{2} \ln \frac{1+r}{1-r} = (\ln(1+r) - \ln(1-r))/2. \quad (8)$$

This transformation stabilizes the variance and can be used to construct a confidence interval (CI) using standard normal theory, defined (in z scale) as

$$\left( z_{min} = z - \frac{q}{\sqrt{t-3}}, z_{max} = z + \frac{q}{\sqrt{t-3}} \right), \quad (9)$$

where  $q$  is the desired normal quantile and  $T$  is the length of the time series. The CI can be transformed back into the original scale using the inverse transformation:

$$r(z) = \frac{e^{2z} - 1}{e^{2z} + 1}, \quad (10)$$

leading to

$$CI = (r(z_{min}), r(z_{max})). \quad (11)$$

The correlation (Eqs. 2, 3, and 4) and cross-correlation distances (Eqs. 5, 6, and 7) can be adapted to consider the statistical test given a significance level  $\alpha$ . These adaptations consist of substituting the correlation function  $r(X, Y)$  in the equations with a binary

$$r_{test}(X, Y) = \begin{cases} 0 & \text{if } r(X, Y) \in CI \text{ } (H_0 \text{ not discarded}) \\ 1 & \text{otherwise } (H_1 \text{ holds}). \end{cases} \quad (12)$$

### *Distances based on information theory*

Mutual information (MI) is one of the most common measures used to detect non-linear relations between time series. The MI measures the average reduction in uncertainty about  $X$  that results from learning the value of  $Y$  or vice versa (MacKay 2003). The concept of information is directly linked to Shannon's entropy, which captures the "amount of information" in a time series. Given a discrete time series  $X$  and its probability mass function  $P(X)$ , the marginal entropy  $H(X)$  is defined as

$$H(X) = - \sum_{i=1}^t P(x_i) \log P(x_i). \quad (13)$$

The conditional entropy  $H(X|Y)$  measures how much uncertainty  $X$  has when  $Y$  is known.

$$H(X|Y) = - \sum_{i,j} p(X_i, Y_j) \log \frac{p(X_i, Y_j)}{p(Y_j)} \quad (14)$$

A histogram can be used to estimate the probabilities. The number of bins in the histogram directly influences the entropy results and should be chosen accordingly. Some common options to estimate the number of bins are the Sturges's, Scott's, and Freedman-Diaconis criteria (Wand 1997).

The mutual information (MI) can be estimated using Eq. 15.

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \quad (15)$$

The mutual information has only a lower bound (zero), which makes it sometimes difficult to interpret the results. Some normalization methods can limit the MI to the interval  $[0, 1]$  to make the comparison easier. The normalized mutual information (NMI) can be defined as

$$NMI(X, Y) = I(X, Y)/U, \quad (16)$$

where  $U$  is the normalization factor. Common normalization approaches are either the half of the sum of the entropies  $U = 0.5H(X)H(Y)$ , minimum entropy  $U = \min(H(X), H(Y))$ , maximum entropy  $U = \max(H(X), H(Y))$ , or the squared root of the entropies product  $U = \sqrt{H(X)H(Y)}$  (Vinh *et al.* 2010). The NMI distance is

$$d_{\text{nmi}}(X, Y) = 1 - NMI(X, Y). \quad (17)$$

Another common distance function based on MI is the Variation of Information (VoI) (Eq. 18) (Vinh *et al.* 2010). VoI, different from the MI, is a true metric.

$$d_{\text{voi}}(X, Y) = H(X) + H(Y) - 2I(X, Y) \quad (18)$$

The maximal information coefficient (MIC) is an algorithm that searches for the largest MI between two series (Reshef *et al.* 2011). MIC considers that if there exists a relationship between two time series, then it is possible to draw a grid on the scatter plot of these two series that encapsulates that relationship. MIC automatically finds the best grid and returns the largest possible MI.

### *Dynamic Time Warping*

The Dynamic Time Warping (DTW) (Berndt and Clifford 1994; Giorgino 2009) is an elastic distance function that uses a warping path to align two time series and then compare them. Different from lock-step measures (such as the euclidean distance), elastic measures use a warping path defining a mapping between the two series (Fig. 1). The optimal path minimizes the global warping cost. The DTW distance  $d_{\text{DTW}}(X, Y) = dtw(i = t, j = t)$  between two time series  $X$  and  $Y$  (both of length  $T$ ) can be defined using dynamic programming. The total distance is the cumulative distance calculated from the recurrence relation  $dtw(i, j)$ ,

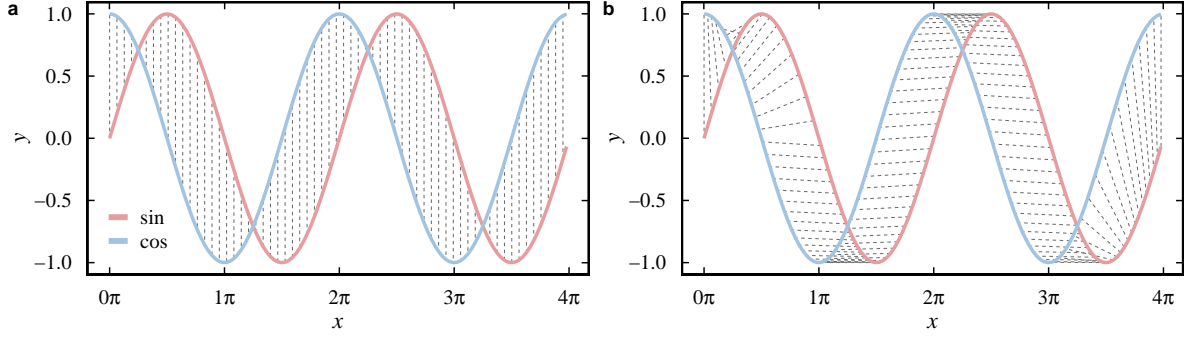


Figure 1: Comparison between lock-step and elastic measures. The figure shows a sine (red) and a cosine (blue) time series. The gray dashed lines show the comparison of the values in (a) a traditional lock-step distance such as the euclidean distance and (b) in an elastic one as in DTW.

defined as:

$$d_{\text{dtw}}(X, Y) = dtw(i = T, j = T) = \begin{cases} \infty & \text{if } i = 0 \text{ xor } j = 0 \\ 0 & \text{if } i = j = 0 \\ |X_i - Y_j| + \min \begin{cases} dtw(i - 1, j) \\ dtw(i, j - 1) \\ dtw(i - 1, j - 1) \end{cases} & \text{otherwise} \end{cases} \quad (19)$$

### Event-based distances

The idea behind event-based distance functions is to compare marked time points where some events occur in the time series. The definition of an event is specific to the domain where the time series comes from. For example, Fig. 2-a illustrates two ECG time series  $X$  and  $Y$  where events (black dots) can be defined as some local maximum values as the T wave representing the ventricular repolarization during the heartbeat. Another example is the concept of extreme events of precipitation that can be defined as the days with rainfall sums above 99th percentiles of wet days during rain season (Boers *et al.* 2019). Common approaches used to define events are local maxima and minima, values higher or lower than a threshold, values higher or lower percentiles, or outliers in the series. An event time series  $X^{ev}$  is here defined as a binary time series composed of ones and zeros representing the presence or absence of an event respectively in a time series  $X$ . Fig. 2-b and c illustrate these events time series for the ECG time series in Fig. 2-a. Another common representation is the events sequence  $X' = \{t_1, t_2, \dots, t_N\}$  that defines the time indices where each event occur.

Event-based distance functions try to find co-occurrences or synchronization between two event time series. The general aim is to count pairs of events in both time series that occur at the same time window of length. For example, considering the five events (black dots) in Fig. 2-a, Fig 2-b shows the co-occurrence of events from  $X^{ev}$  to  $Y^{ev}$  considering a time window. For the five events, only the last one does not co-occur. Fig 2-c shows the inverse process that can or not be equal depending on the specific distance function. Thus, functions

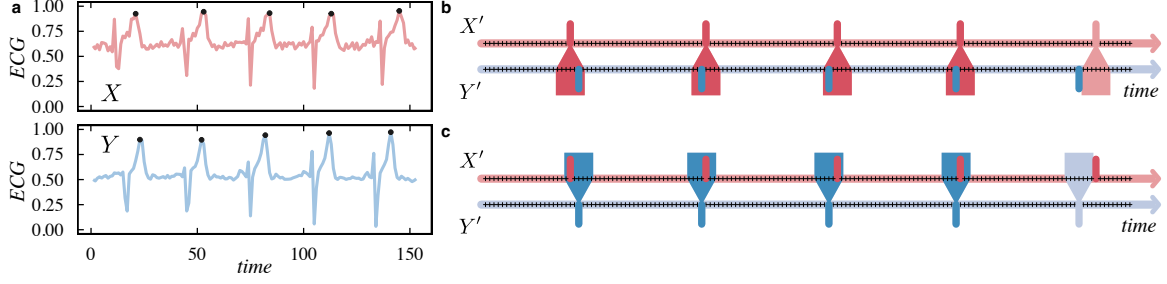


Figure 2: Intuition behind event-based distance functions. (a) Two ECG time series  $X$  and  $Y$  are depicted in red (top) and blue (bottom), respectively. Data were extracted and adapted from (Goldberger *et al.* 2000; Liu *et al.* 2016). Black dots represent marked events in both time series. (b and c) Segments in red ( $X^{ev}$ ) and blue ( $Y^{ev}$ ) represent the events (black dots) in  $X$  and  $Y$ , respectively. (b) For each event in  $X^{ev}$ , the distance function tries to find another event in  $Y^{ev}$  considering a time window ( $\tau = \pm 3$ ), illustrated by the red and blue pentagons. (c) The same search can be performed the other way around.

can be symmetrical or not (directed). Symmetrical distances consider co-occurrences from  $X$  to  $Y$  but also from  $Y$  to  $X$ . Directed measures only consider the events in a precursor time series  $X$  compared to another one  $Y$ , but not the other way around.

Quian Quiroga *et al.* (2002) proposed a method to measure the synchronization between two event time series. Considering two event sequences  $X' = \{t_1, \dots, t_{N_X}\}$  and  $Y' = \{t_1, \dots, t_{N_Y}\}$ , and a time lag  $\tau$ , this method counts the number of times an event appears in  $X'$  shortly after it appears in  $Y'$  considering using the following equation

$$c^\tau(X'|Y') = \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} J_{ij}^\tau, \quad (20)$$

where  $J_{ij}^\tau$  is defined as

$$J_{ij}^\tau = \begin{cases} 1 & , \text{ if } 0 < t_i^X - t_j^Y \leq \tau \\ 1/2 & , \text{ if } t_i^X = t_j^Y \\ 0 & , \text{ otherwise} \end{cases} \quad (21)$$

where  $t_i^X$  and  $t_j^Y$  are the time indices  $i$  and  $j$  from  $X'$  and  $Y'$  respectively. Analogously,  $c^\tau(Y'|X')$  can also be calculated using Eq. 20. The symmetrical event-based distance between  $X'$  and  $Y'$ , can be calculated using Eq. 22, where  $d_q(X', Y') = 0$  indicates full synchronization.

$$d_{es}^{sy}(X', Y') = 1 - \frac{c^\tau(X'|Y') + c^\tau(Y'|X')}{\sqrt{N_X N_Y}}. \quad (22)$$

The asymmetrical version counts only the number of events in  $X'$  that precede those in  $Y'$  (or the opposite in an analogous way), and it is originally defined in the interval  $[-1, 1]$ . To standardize the interval with the other distances in this paper, Eq. 23 measures the asymmetrical distance where zero means that all events in  $X'$  precede those in  $Y'$ , and one no synchronization.

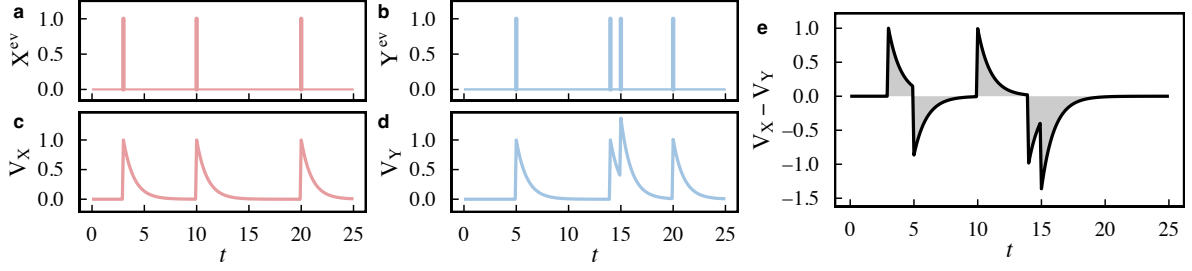


Figure 3: General idea behind the van-Rossum distance (v. Rossum 2001). Two event time series (a)  $X^{ev}$  and (b)  $Y^{ev}$  where vertical segments represent the events in both time series. The von-Rossum distance transforms each event into a real function illustrated respectively in (c) and (d). (e) Considering the difference between the two transformed time series  $V_X - V_Y$ , the von-Rossum distance corresponds to the area under this curve (light gray).

$$d_{es}^{as}(X', Y') = 1 - \frac{c^\tau(Y'|X') - c^\tau(X'|Y') + \sqrt{N_X N_Y}}{2\sqrt{N_X N_Y}} \quad (23)$$

The authors also propose a local time window  $\tau_{ij}$  when event rates changes in the time series. The counting function (Eq. 20) can be adapted to a local time window by replacing  $\tau$  by

$$\tau_{ij} = \min\{t_{i+1}^X - t_i^X, t_i^X - t_{i-1}^X, t_{j+1}^Y - t_j^Y, t_j^Y - t_{j-1}^Y\}/2. \quad (24)$$

Boers *et al.* (2019) introduce a parameter  $\tau_{\max}$  that limits the maximum local time window (Eq. 24).

The van Rossum distance (v. Rossum 2001) is another method used to find synchronizations between events. The main idea is to transform each event into a real function and compare them, as illustrated in Fig. 3. The first step consists on mapping an event sequence  $X' = \{t_1, \dots, t_N\}$  into a function  $V(X') = 1/N \sum_{i=1}^N h_\tau(t - x_i)u(t - x_i)$ , where  $h_\tau(t)$  is a filtering function. Examples of filtering functions include the Gaussian  $h_\tau^g(t) = \exp(-t^2/(2\tau^2))/\sqrt{2\pi\tau^2}$  and the laplacian  $h_\tau^l(t) = \exp(-|t|/\tau)/(2\tau)$  kernels. The parameter  $\tau$  defines the co-occurrence time scale. The van Rossum distance can be calculated as

$$d_{vr}(X', Y') = \sqrt{\int_0^\infty [V(X') - V(Y')]^2 dt}. \quad (25)$$

One common approach to test the significance of event-based distance functions considers surrogate data (Boers *et al.* 2019; Ferreira *et al.* 2021b). The idea is to generate artificial event sequences by shuffling the events in the two original time series and then measuring their distance. After repeating this process a considerable high number of times, it is possible to construct a distribution of distances and use it as a confidence level for the comparison.

## 2.2. Transforming a set of time series into a network

A network or a graph (Bondy and Murty 2008; Barabási and Pósfai 2016),  $G(V, E)$  is composed by a set of  $n$  nodes (also called vertices)  $V$  and a set of  $m$  links (also called edges)  $E$  connecting two nodes. The most common representation for a network is the adjacency



matrix  $A$  whose entry  $a_{ij} = 1$  if the nodes  $i$  and  $j$  are connected or  $a_{ij} = 0$  otherwise. A weighted network can also be represented by  $A$ . In this case, each entry  $a_{ij} = w_{ij} \in \mathbb{R}_{\neq 0}$  corresponds to the weight  $w_{ij}$  between nodes  $i$  and  $j$ , or  $a_{ij} = 0$  if there is no link.

A set of time series  $\mathcal{X}$  can be transformed into a network by calculating the distance between all pairs of time series. These distances are represented by the distance matrix  $D$  whose position  $d_{ij}$  stores the distance between the time series  $i$  and  $j$ . The network construction process consists of transform the distance matrix  $D$  into an adjacency matrix  $A$  where each node represents a time series. The most common building approaches are (Silva and Zhao 2018; Qiao *et al.* 2018):

- *k-NN network*: In a  $k$  nearest neighbors network ( $k$ -NN), each node (representing a time series) is connected to the  $k$  other ones with the shortest distances. Thus, the construction process requires finding for each row  $i$  of  $D$  the  $k$  other elements with shortest distances, where  $k \in \mathbb{W}$  is a parameter defined by the user. Approximation algorithms speed up the construction process (Arya *et al.* 1998; Chen *et al.* 2009).
- *$\epsilon$ -NN network*: In a  $\epsilon$  nearest neighbors network ( $\epsilon$ -NN), (Fig. 4-b), each node is connected to all the other ones if their distances are shorter than  $\epsilon \in \mathbb{R}_+$ , defined by the user. An advantage of the  $\epsilon$ - over  $k$ -NN networks is that the  $\epsilon$  permits a finer in the threshold choice. Another advantage is the fact that  $\epsilon$  can restrict the connections to only strong ones, while the  $k$ -NN will always connect nodes regardless if the  $k$  shortest distances are small or high. Similar to  $k$ -NN networks, approximations can speed up the construction process (Arya *et al.* 1998; Chen *et al.* 2009).
- *Weighted network*: Instead of limiting the number of links, a weighted network (Fig. 4-c) can be constructed from  $D$  by simply connecting all pairs of nodes and using their distances to set weights. In general, link weights have the opposite definition of a distance, which means that the shorter the distance between two time series, the stronger the link connecting the respective nodes. Considering that  $D$  is in the interval  $[0,1]$ , the weighted adjacency matrix can be defined as  $A = 1 - D$ . Conversely, if  $D \in [0, +\infty]$ ,  $D$  can be first normalized between  $[0,1]$  ( $D_{\text{norm}}$ ) and then consider  $A = 1 - D_{\text{norm}}$ .
- *Networks with significant links*: In many applications, it is important to minimize the number of spurious links in the network. One way of minimizing these spurious links is by considering statistical tests for the distance functions. In other words, this network construction process connects all pairs of nodes only if their distance is statistically significant. For example, the significance of the Pearson correlation coefficient can be tested using the z-transformation (Eq. 9) described in Sec. 2.1. This network construction process can be seen as a special case of  $\epsilon$ -NN network where  $\epsilon$  is a threshold for significant links. It is important to remember that this building approach minimizes the number of spurious links but does not guarantee that they will not appear.

Networks can be static or temporal (Holme and Saramäki 2012; Kivelä *et al.* 2014). A static network  $G(V, E)$  uses the whole historical data in a time series to construct a single network. Temporal networks divide the historical period into time windows that are used to construct layers in a temporal (multilayered) network. A temporal network is a sequence of static networks ordered in time. Each layer can be seen as a snapshot of the time window. The temporal network construction process requires the computation of a set of distance matrices

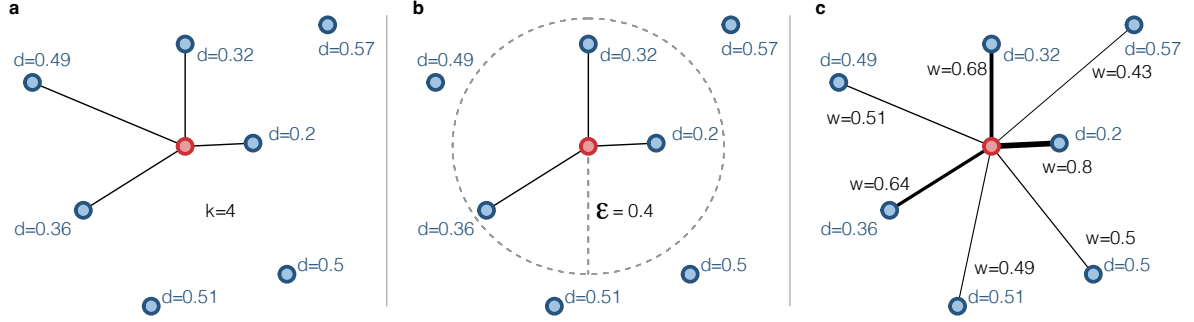


Figure 4: Geometrical intuition behind three different network construction processes. Each time series is represented by a blue node in the network, except the reference time series represented in red. The figures show the distances from the reference time series to all the other ones (a) In a  $k$ -NN ( $k=4$ ) network, the reference node is connected to the four other ones with the shortest distances. (b) In a  $\epsilon$ -NN ( $\epsilon = 0.4$ ) network, the reference node is linked to all the nodes whose distance is shorter than  $\epsilon$  (within the dashed circle). (c) In the weighted graph, the reference node is connected to all the other ones, and the link weight is ( $w = 1 - d$ ). The link thickness is proportional to the weight.

$\mathcal{D} = \{D_1, \dots, D_W\}$ , one for each of the  $W$  windows. A time window is defined by the user and has a length  $\Delta t$  and a step  $s$  that defines the number of values that are included and excluded in each window. The temporal network construction consists of transforming each distance matrix  $D \in \mathcal{D}$  into a network layer using the same process used for a static network previously described in this section. One of the main advantages of this approach is to make possible the temporal analysis (evolution) of the modeled system.

The network construction process has a combinatorial nature. Considering a set of  $n$  time series, the whole construction process requires  $\binom{n}{2}$  distance calculations in a static network or  $W \cdot \binom{n}{2}$  in a temporal network, where  $W$  is the number of layers. The transformation from the distance matrix to the adjacency matrix requires searches for pairs of shortest nodes, a task that is much faster than the distance calculations. For this reason, the conversion step ( $D \rightarrow A$ ) can be neglected in the total time complexity is  $O(\binom{n}{2})$ . This means that the number of distance computations can increase very fast for very large time series data sets, making the network modeling approach infeasible.

### 2.3. Transforming a single time series into a network

This section presents methods to transform a single time series into a network (Zou *et al.* 2019; Silva *et al.* 2021).

#### *Proximity networks*

The most straightforward method to perform this transformation consists of breaking the time series into time windows and using the same approach described for multiple time series (Sec. 2.2). Each time window is considered a time series and the network construction process requires calculating a distance matrix  $D$  between all time windows using a distance function. Then,  $D$  can be transformed into an adjacency matrix using methods such as  $k$ -NN and  $\epsilon$ -NN networks (Silva and Zhao 2018). The time window size should be carefully chosen. For

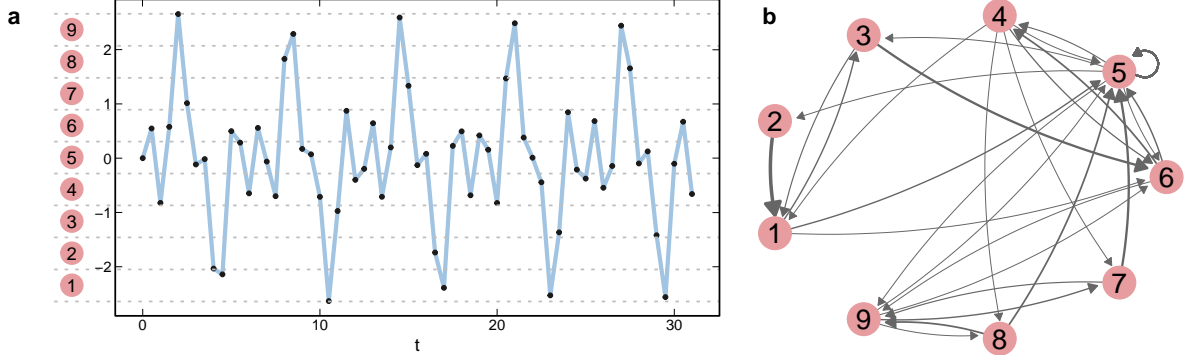


Figure 5: Transition networks. (a) Example of time series sampled from  $y(t) = \sin(t) + \sin(4t) - \sin(2t)$ . This example uses nine equally-spaced bins divided by the dashed lines. Each bin is represented by a node depicted in red. (b) The resulting transition network is illustrated with edge widths proportional to the transition probability.

example, time windows too small can be problematic when calculating correlation distances. For this reason, this simple construction procedure may not be suitable for short time series because it could generate small time windows.

#### Transition networks

Transition networks map time series into symbols, represented by nodes and edges to link transitions between symbols. [Campanharo et al. \(2011\)](#) proposed a straightforward way of defining the symbols, which consists of first dividing a time series into equally-spaced bins. Each bin corresponds to a symbol that is mapped to a node in the network. Each time series value is mapped to a sequence of symbols according to the bins they are inserted. Then, for every pair of consecutive symbols, a directed link is established between the corresponding nodes. Two equal consecutive symbols lead to self-loops in the network. Edge weights count the number of times that a transition between the two symbols occurs. After the construction process, the weights of the outgoing links from each node can be normalized and interpreted as a transition probability between symbols. In this case, the network can be seen as a Markov Chain. Fig. 5 presents an example of a transition network.

#### Visibility graphs

[Lacasa et al. \(2008\)](#) proposed a method called visibility graph that transforms a time series into a network. The idea is to represent each value in a time series by a bar in a barplot, as illustrated in Fig. 6. Then, each value becomes a node and links connect them if they can “see” each other. In the original definition, called natural visibility graph (NVG), the concept of visibility is defined as a line in any direction that does not cross any other bar. Considering a time series  $X = \{X_i, \dots, X_N\}$  with  $N$  values and its natural visibility graph  $NVG^X = \{v_i, \dots, v_N\}$ , two nodes  $v_i$  and  $v_j$  are connected if

$$X_k < X_i + (X_j - X_i) \frac{k - i}{j - k} \quad \forall k \in \{i < k < j\}. \quad (26)$$

Other works adapted or restricted the idea of visibility. The horizontal visibility graph (HVG)

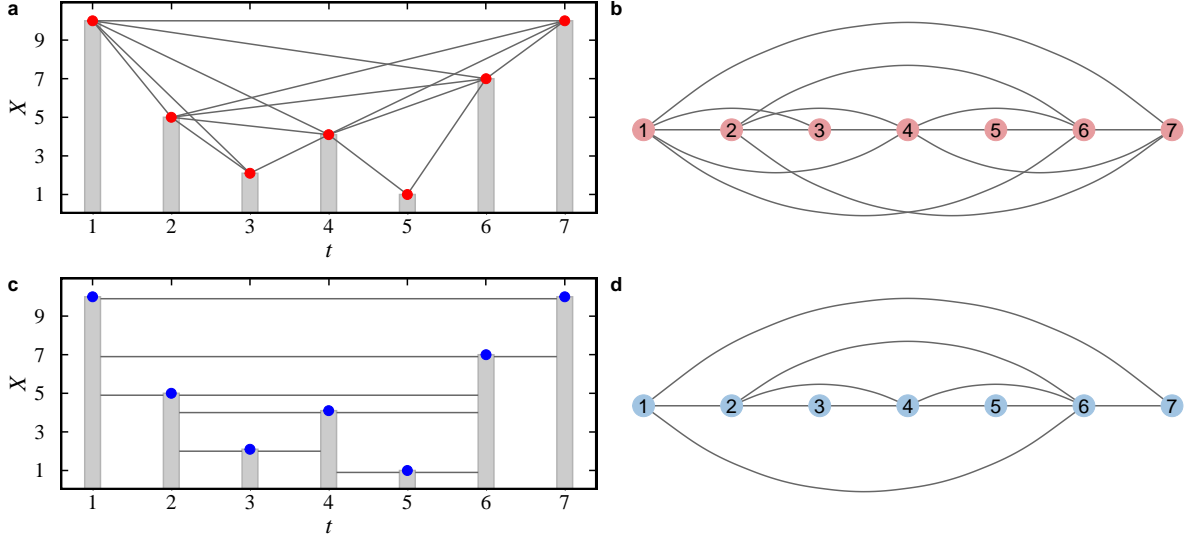


Figure 6: Visibility graph construction. (a and c) An example of time series with values represented by the bars and points. Gray lines connect “visible” values as defined in the (a) natural (red) and (c) horizontal (blue) visibility graphs. The resulting natural (b) and horizontal (d) visibility graphs.

(Luque *et al.* 2009) defines visibility as horizontal lines between bars (Fig. 6 c and d). In this case, two nodes  $v_i$  and  $v_j$  from  $HVG^X$  are connected if

$$X_k < X_i, X_j \quad \forall k \in \{i < k < j\}. \quad (27)$$

Threshold values can limit the maximum temporal distance between values (Ting-Ting *et al.* 2012). Visibility graphs can also be directed, representing the temporal order. Another important concern is the time complexity of the construction process. The original algorithms to construct NVGs and HVGs have quadratic time complexity  $O(n^2)$ , where  $n$  is the number of elements in the series. Further improvements can reduce this time complexity to  $O(n \log n)$  (Lan *et al.* 2015).

### Recurrence networks

The transformation approach proposed by Donner *et al.* (2010) relies on the concept of recurrences in the phase space. The general idea consists of interpreting the recurrence matrix of a time series as an adjacency matrix. Nodes represent states in the phase-space and links connect recurrent ones. A recurrence matrix connects different points in time if the states are closely neighbored in phase space. Considering a time series  $X$  and its  $m$ -dimensional time delay embedding  $\vec{x} = (X_t, X_{t+\tau}, \dots, X_{t+(m-1)\tau})$  where  $\tau$  is the delay, a recurrence matrix can be defined as

$$R_{ij}(\epsilon) = \Theta(\epsilon - d(\vec{x}_i - \vec{x}_j)), \quad (28)$$

where  $\Theta(\cdot)$  is the Heaviside function,  $d(\cdot)$  is a distance function (e.g., Euclidean or Manhattan), and  $\epsilon$  is a threshold distance that defines close states. A recurrence matrix  $R(\epsilon)$  can then be

transformed into an adjacency matrix  $A$  of a network as:

$$A_{ij} = R_{ij}(\epsilon) - \delta_{i,j}, \quad (29)$$

where  $\delta_{i,j}$  is the Kronecker delta introduced to remove self-loops. The resulting network is unweighted, undirected, and without self-loops

### 3. The **ts2net** Package

The goal of the **ts2net** package is to provide methods to model a single or a set of time series into a network in R. It implements time series distance functions, parallel distance calculations, network construction methods, and some useful functions. Some functions are fully implemented, while others are adapted from other packages. Thus, another goal of the **ts2net** package is to put together useful methods from other packages.

The **ts2net** package is available on the Comprehensive R Archive Network (CRAN) and can be installed with the following command:

```
R> install.packages("ts2net")
```

Another option is to install **ts2net** from GitHub using the `install_github()` function from either **remotes** or **devtools** packages:

```
R> install.packages("remotes") # if not installed
R> remotes::install_github("lnferreira/ts2net")
```

#### 3.1. Implementation aspects

The **ts2net** package relies on some other important packages. One of the most important ones is **igraph**, which is used to model networks (Csardi and Nepusz 2006). The **igraph** package provides many different graph and network analysis tools, which can be used to extract information from the modeled time series. The **igraph** objects can be easily transformed or exported to other formats such as graphml, gml and edge list. Besides, other network analysis packages commonly accept these objects. Examples are the **sna** (Butts 2020), **tsna** (Bender-deMoll and Morris 2021), and **muxViz** (De Domenico *et al.* 2014).

The network construction process normally requires a high number of calculations. For this reason, the most important functions in **ts2net** are implemented in parallel using multi-core programming. The parallelization was implemented using the `mclapply()` function from the **parallel**, which comes with R. This implementation uses forks to create multiple processes, an operation that is not available in Windows systems. It means that, on Windows, all functions in **ts2net** with `num_cores` can only run with one core.

Besides the multicore parallelization, the **ts2net** was also constructed to be run in supercomputers and computer clusters. The idea is to divide the distance calculations into jobs that could be calculated in parallel and then merged together to construct the distance matrix. The `ts_dist_dir()` and `ts_dist_dir()` functions make this process easier.

### 3.2. Distance calculation

Functions to calculate the distance matrix:

- `ts_dist()`: Calculates all pairs of distances and returns a distance matrix  $D$ . This function receives a list of time series which can either have or not have the same length. It runs serial or in parallel (except in Windows) using `mclapply()` from **parallel** package. This function receives a distance function as a parameter. **Ts2net** provides some distance functions (functions starting with `tsdist_...`). Other distances can be easily implemented or adapted to be used within this function.
- `ts_dist_part()`: Calculate distances between pairs of time series (similarly to `ts_dist()`) in part of a list. This function is particularly useful to run in parallel as jobs in supercomputers or computer clusters (HPC). Instead of having a single job calculating all distances (with `ts_dist()`), this function permits the user to divide the calculation into different jobs. The result is a data frame with the distances  $d_{ij}$  and indexes  $i$  and  $j$ . To construct the distance matrix  $D$ , the user can merge all the results using the functions `dist_parts_merge()` or `dist_parts_file_merge()`.
- `ts_dist_part_file()`: This function works similar as `ts_dist_part()`. The difference is that it reads the time series from RDS files in a directory. The advantage of this approach is that it does not load all the time series in memory but reads them only when necessary. This means that this function requires much less memory and should be preferred when memory consumption is a concern, e.g., huge data set or very long time series. The disadvantage of this approach is that it requires a high number of file read operations which considerably takes more time during the calculations. The user can merge all the results using the functions `dist_parts_merge()` or `dist_parts_file_merge()`.

List of distance functions:

- `tsdist_cor()`: Absolute (Eq. 2), positive (Eq. 3) or negative (Eq. 4) correlation distances. This function optionally test the significance for the correlation using the Fisher z-transformation (Eq. 8) (Asuero *et al.* 2006).
- `tsdist_ccf()`: Absolute (Eq. 5), positive (Eq. 6), or negative (Eq. 7) cross-correlation distances.
- `tsdist_dtw()`: Dynamic Time warping (DTW) distance (Sec. 2.1.3). This function is a wrapper for the `dtw()` function from the **dtw** package (Giorgino 2009).
- `tsdist_nmi()`: Normalized mutual information distance (Eq. 17). The implementation uses functions from the **infotheo** package (Meyer 2014).
- `tsdist_voi()`: Variation of information distance (Eq. 18). The implementation uses functions from the **infotheo** package (Meyer 2014).
- `tsdist_mic()`: Maximal information coefficient (MIC) distance. This function transforms the `mine()` function from the **minerva** package (Albanese *et al.* 2012) into a distance.

- `tsdist_es()`: The events synchronization distance (Eq. 22) proposed by Quian Quiroga *et al.* (2002). This function can also use the modification proposed by Boers *et al.* (2019) ( $\tau_{\max}$ ). This function optionally test the significance for the events synchronization by shuffling the events in the time series.
- `tsdist_vr()`: Van Rossum distance. This function uses the `fmetric()` function from the `mmpp` package (Hino *et al.* 2015). It optionally test the significance for the events synchronization by shuffling the events in the time series.

Other distances can be easily applied in the **ts2net**. The only two requirements are that the distance function should accept two arrays and it must return a distance. The packages **TSclust** (Montero and Vilar 2014) and **TSdist** (Mori *et al.* 2016) provide many other distance functions that can be used to create network with **ts2net**. For example, the euclidean distance (`diss.EUCL()` from **TSclust**) can be used to construct the distance matrix:

```
R> install.packages("TSclust")
R> library(TSclust)
R> D <- ts_dist(ts_list, dist_func=diss.EUCL)
```

It is important to remember that the parameter `dist_func` receives a function, so no parenthesis is necessary (`dist_func=diss.EUCL()` is an error).

Another possibility is to implement specific functions and apply them. For example, the following code snippet implements a distance function that compares the mean values of each time series and applies it to calculate the distance matrix  $D$ :

```
R> ts_dist_mean <- function(ts1, ts2) abs(mean(ts1) - mean(ts2))
R> D <- ts_dist(ts_list, dist_func=ts_dist_mean)
```

### 3.3. Network construction methods

Multiple time series into a network:

- `net_knn()`: Constructs a  $k$ -NN network from a distance matrix  $D$ .
- `net_knn_approx()`: Creates an approximated  $k$ -NN network from a distance matrix  $D$ . This implementation may omit some actual nearest neighbors, but it is considerably faster than `net_knn()`. It uses the `kNN()` function from the **dbscan** package.
- `net_enn()`: Constructs a  $\epsilon$ -NN network from a distance matrix  $D$ .
- `net_enn_approx()`: Creates an approximated  $\epsilon$ -NN network from a distance matrix  $D$ . This implementation may omit some actual nearest neighbors, but it is considerably faster than `net_enn()`. It uses the `frNN()` function from the **dbscan** package.
- `net_weighted()`: Creates a full weighted network from a distance matrix  $D$ . In this case,  $D$  should be in the interval  $[0, 1]$  (see `dist_matrix_normalize()`). The link weights in the resulting network is  $w_{ij} = 1 - d_{ij}$ .
- `net_significant_links()`: Construct a network with significant links.



A single time series into a network:

- `ts_to_windows()`: Extract time windows from a time series. This function can be used to construct a network from time windows. It returns a list of windows that can be used as input for `ts_dist()` to construct a distance matrix  $D$ .
- `tsnet_qn()`: Creates transition (quantile) networks.
- `tsnet_vg()`: Creates natural and horizontal visibility graphs. This function can be run in parallel.
- `tsnet_rn()`: Creates a recurrence network from a time series. This function uses the `rqa()` function from the **nonlinearTseries** package to obtain the recurrence matrix.

### 3.4. Useful functions

- `dist_matrix_normalize()`: normalizes (min-max) a distance matrix  $D$  between zero and one.
- `dist_percentile()`: Returns the distance value that corresponds to the desired percentile. This function is useful when the user wants to generate networks with different distance functions but with the same link density.
- `ts_dist_dirs_merge()`: `ts_dist_dir()` and `ts_dist_dir()` calculate parts of the distance matrix  $D$ . This function merges the results and constructs a distance matrix  $D$ .
- `tsdist_file_parts_merge()`: Merges the distance parts as `ts_dist_dirs_merge()`, but reading them from files.
- `events_from_ts()`: Function to extract events from a time series. For example, it can return the desired percentile of highest or lowest values. This function is particularly useful for event-based distance functions.

### 3.5. Data

- `dataset_sincos_generate()`: Generates a data set with sine and cosine time series with arbitrary noise.
- `random_ets()`: Creates random event time series with uniform probability.
- `us_cities_temperature_df`: A data set containing the temperature in 27 US cities between 2012 and 2017. This data set was adapted from the original ([Beniaguev 2017](#)) for simplicity reasons. Different from the original, this data set comprehends only cities in the US, grouped by month (mean value) temperature and without days with missing data. The data set is stored in a data frame with 61 rows and 28 variables where each column (except date), corresponds to the mean temperature values (°C) for each city.



- `us_cities_temperature_list`: Similar to `us_cities_temperature_df`, but in a list, which is the required format for distance calculations (e.g. `ts_dist()`).

### 3.6. Source code

The **ts2net** is open-source and free to use according to the MIT License (<https://github.com/lnferreira/ts2net/blob/main/LICENSE>). All the code is available on Github at <https://github.com/lnferreira/ts2net>.

## 4. Applications

This section presents how the **ts2net** package can be used to transform a set or a single time series into networks. Two real data sets were used as examples. The goal here is not to present complicated applications and deep analyses but straightforward and didactic examples of how to use networks to model time series.

### 4.1. Transforming a set of time series into a network

This section presents a network modeling example using real data. The data set is composed of monthly temperature data from 27 cities in the US between 2012 and 2017, illustrated in Fig. 7-a. The version presented here was extracted and adapted from Kaggle (Beniaguev 2017) for simplicity reasons by taking the monthly averages and keeping only US cities.

The following code calculates the distance matrix (Fig. 7-b) using DTW, finds the  $\epsilon$  that corresponds to the 30% of the shortest distances, constructs a  $\epsilon$ -NN network (Fig. 7-c), and find communities using the algorithm proposed by Girvan and Newman (2002):

```
R> D <- ts_dist(us_cities_temperature_list, dist_func = tsdist_dtw)
R> eps <- dist_percentile(D, percentile = 0.3)
R> net <- net_enn(D, eps)
R> cluster_edge_betweenness(net)
```

```
IGRAPH clustering edge betweenness, groups: 2, mod: 0.33
```

```
+ groups:
```

```
$`1`
```

```
[1] "Portland"      "Seattle"      "Albuquerque"  "Denver"
[5] "Kansas City"  "Minneapolis" "Saint Louis"  "Chicago"
[9] "Nashville"    "Indianapolis" "Detroit"      "Charlotte"
[13] "Pittsburgh"   "Philadelphia" "New York"     "Boston"
```

```
$`2`
```

```
[1] "San Francisco" "Los Angeles"  "San Diego"
[4] "Las Vegas"     "Phoenix"      "San Antonio"
[7] "Dallas"        "Houston"      "Atlanta"
```

```
+ ... omitted several groups/vertices
```

Fig. 7-c illustrates the resulting network where nodes are conveniently placed according to their geographical position. Node colors represent the two communities that cluster together colder (blue nodes) and hotter (red nodes) cities.

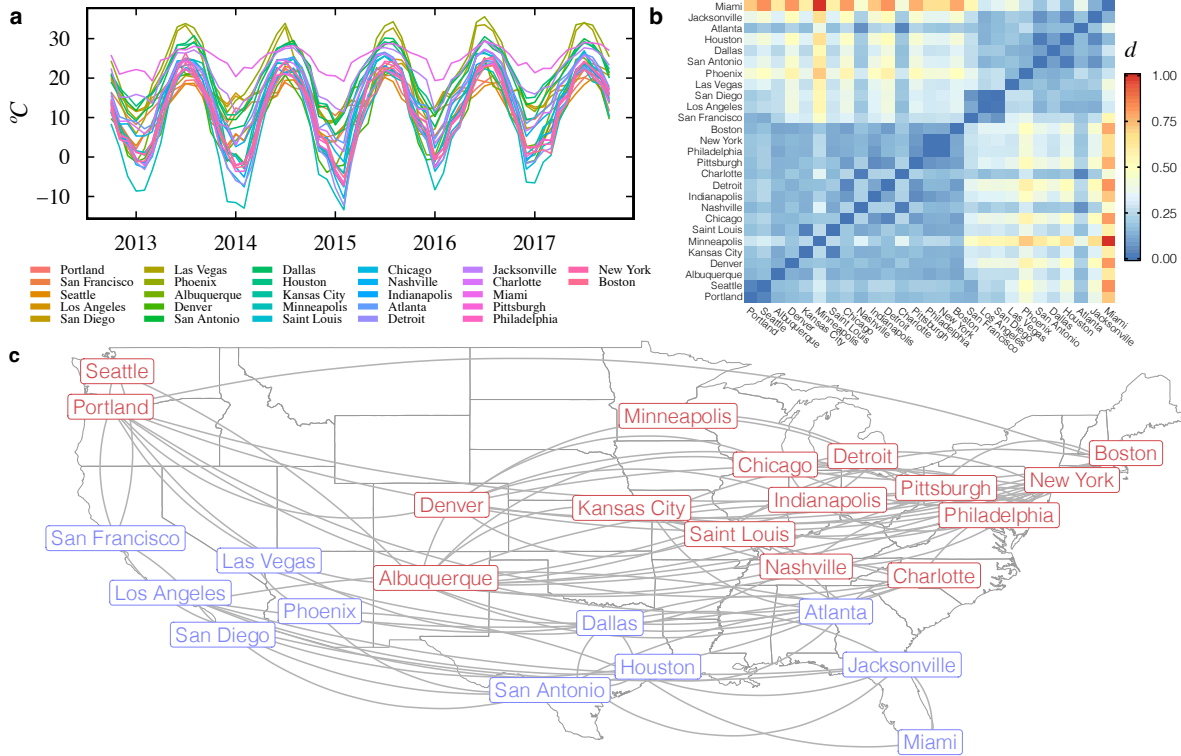


Figure 7: Transforming time series into a network using **ts2net**. (a) The historical temperature of 27 cities in the US. (b) The distance matrix  $D$  (normalized DTW) for the data set. (c) The  $\epsilon$ -NN network was constructed using 30% of the shortest distances. Node colors represent communities.

## 4.2. Transforming a single time series into a network

This section presents how a time series extracted from real data can be modeled as networks. The time series with the monthly atmospheric concentration of carbon dioxide (ppm) measured in the Mauna Loa Observatory between 1959 and 1997 (Keeling *et al.* 2005), illustrated in Fig. 8a. This data set was mainly chosen by its simplicity and accessibility (available with the base installation of R).

The following code generates a time-window network (**net\_w**) with width 12 and one-value step, a visibility graph (**net\_vg**), a recurrence network (**net\_rn**) with  $\epsilon = 5$ , and a transition network (**net\_qn**) using 100 equally spaced bins using the **ts2net** package.

```
R> net_w = as.vector(co2) />
  ts_to_windows(width = 12, by = 1) />
  ts_dist(cor_type = "+") />
  net_enn(eps = 0.25)
```

```

R> net_vg = as.vector(co2) />
      tsnet_vg()
R> net_rn = as.vector(co2) />
      tsnet_rn(radius = 5)
R> net_qn = as.vector(co2) />
      tsnet_qn(breaks = 100)

```

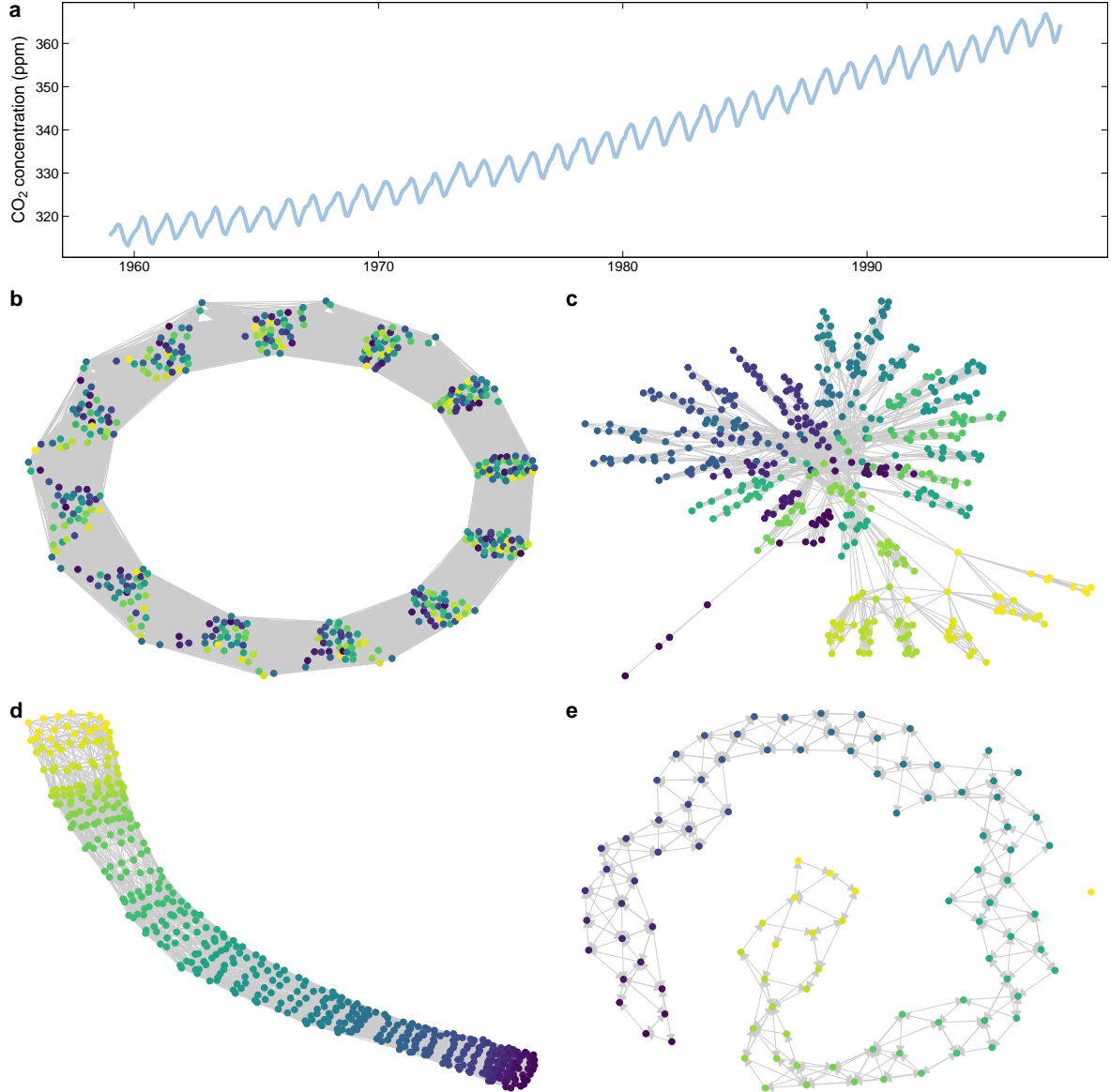


Figure 8: Transforming a single time series into networks using **ts2net**. (a) The CO<sub>2</sub> concentration time series. (b) Proximity network with time window 12 and one-value step. (c) Natural visibility graph. (d) Recurrence network ( $\epsilon = 5$ ). (e) Transition (quantile) network (100 equally-spaced bins). Node colors represent temporal order (yellow to blue), except in the transition network where colors represent the sequence of lower (yellow) to higher (blue) bins.

Figures 8b-e illustrate the networks resulting from the transformation of the CO<sub>2</sub> time series. From the topology of the resulting networks, it is possible to extract useful information. For example, the time-window network (Fig. 8b) shows 12 groups representing each month. The visibility graph (Fig. 8c) presents many small-degree nodes that represent the values on valleys while a few hubs connect peaks values of CO<sub>2</sub> after 1990 that connect to many other nodes. These hubs appear due to the increasing trend of CO<sub>2</sub> contraction and some slightly higher peaks during the seasons after 1980. The recurrence (Fig. 8d) and transition (Fig. 8e) networks present some local connections caused by the seasons, and a line shape as a result the increasing trend in the CO<sub>2</sub> concentration over the years.

## 5. Final considerations

This paper presents **ts2net**, a package to transform time series into networks. One important drawback when transforming a set of time series into a network is the high computational cost required to construct a distance matrix (combinations of time series), making it unfeasible for huge data sets. To minimize this problem, **ts2net** provides tools to run distance functions in parallel and in high-performance computers via multiple jobs.

The **ts2net**'s goal is not to provide all transformation methods in the literature but the most used ones. It does not mean that this package cannot be extended or customized. For example, as explained in Sec. 3.2, this package accepts other time series distance functions implemented by other packages or the user. Additional modeling approaches can also be incorporated to the package in the future.

## Computational details

The **ts2net** package requires R version 4.1.0 or higher. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/>.

## References

- Albanese D, Filosi M, Visintainer R, Riccadonna S, Jurman G, Furlanello C (2012). “Minerva and minepy: a C engine for the MINE suite and its R, Python and MATLAB wrappers.” *Bioinformatics*, p. bts707.
- Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998). “An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions.” *J. ACM*, **45**(6), 891–923. ISSN 0004-5411. doi:10.1145/293347.293348.
- Asuero AG, Sayago A, González AG (2006). “The Correlation Coefficient: An Overview.” *Critical Reviews in Analytical Chemistry*, **36**(1), 41–59. doi:10.1080/10408340500526766.
- Barabási AL, Pósfai M (2016). *Network science*. Cambridge University Press, Cambridge. ISBN 9781107076266 1107076269.
- Bender-deMoll S, Morris M (2021). *tsna: Tools for Temporal Social Network Analysis*.

- Beniaguev D (2017). “Historical Hourly Weather Data 2012-2017.” <https://www.kaggle.com/datasets/selfishgene/historical-hourly-weather-data>. [Online; accessed 01-June-2022].
- Berndt DJ, Clifford J (1994). “Using Dynamic Time Warping to Find Patterns in Time Series.” In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS’94, pp. 359–370. AAAI Press.
- Boers N, Goswami B, Rheinwalt A, Bookhagen B, Hoskins B, Kurths J (2019). “Complex networks reveal global pattern of extreme-rainfall teleconnections.” *Nature*, **566**(7744), 373–377. doi:10.1038/s41586-018-0872-x.
- Bondy JA, Murty USR (2008). *Graph theory*. Springer.
- Bullmore E, Sporns O (2009). “Complex brain networks: graph theoretical analysis of structural and functional systems.” *Nature Reviews Neuroscience*, **10**(3), 186–198. doi:10.1038/nrn2575.
- Butts CT (2020). *sna: Tools for Social Network Analysis*. R package version 2.6.
- Campanharo ASLO, Simer MI, Malmgren RD, Ramos FM, Amaral LAN (2011). “Duality between Time Series and Networks.” *PLOS ONE*, **6**(8), 1–13. doi:10.1371/journal.pone.0023378.
- Chen J, Fang Hr, Saad Y (2009). “Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection.” *J. Mach. Learn. Res.*, **10**, 1989–2012. ISSN 1532-4435.
- Csardi G, Nepusz T (2006). “The igraph software package for complex network research.” *InterJournal*, **Complex Systems**, 1695.
- De Domenico M, Porter MA, Arenas A (2014). “MuxViz: a tool for multilayer analysis and visualization of networks.” *Journal of Complex Networks*, **3**(2), 159–176. ISSN 2051-1310. doi:10.1093/comnet/cnu038.
- Deza MM, Deza E (2009). “Encyclopedia of distances.” In *Encyclopedia of distances*, pp. 1–583. Springer.
- Donner RV, Zou Y, Donges JF, Marwan N, Kurths J (2010). “Recurrence networks—a novel paradigm for nonlinear time series analysis.” *New Journal of Physics*, **12**(3), 033025. doi:10.1088/1367-2630/12/3/033025.
- Esling P, Agon C (2012). “Time-Series Data Mining.” *ACM Comput. Surv.*, **45**(1). ISSN 0360-0300. doi:10.1145/2379776.2379788.
- Ferreira LN, Ferreira NCR, Macau EEN, Donner RV (2021a). “The effect of time series distance functions on functional climate networks.” *The European Physical Journal Special Topics*, **230**(14), 2973–2998. doi:10.1140/epjs/s11734-021-00274-y.
- Ferreira LN, Hong I, Rutherford A, Cebrian M (2021b). “The small-world network of global protests.” *Scientific Reports*, **11**(1), 19215. doi:10.1038/s41598-021-98628-y.

- Giorgino T (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package.” *Journal of Statistical Software*, **31**(7), 1–24. doi:10.18637/jss.v031.i07.
- Girvan M, Newman MEJ (2002). “Community structure in social and biological networks.” *Proceedings of the National Academy of Sciences*, **99**(12), 7821–7826. doi:10.1073/pnas.122653799.
- Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE (2000). “PhysioBank, PhysioToolkit, and PhysioNet.” *Circulation*, **101**(23), e215–e220. doi:10.1161/01.CIR.101.23.e215.
- Hino H, Takano K, Murata N (2015). “mmp: A Package for Calculating Similarity and Distance Metrics for Simple and Marked Temporal Point Processes.” *The R Journal*, **7**(2), 237–248. doi:10.32614/RJ-2015-033.
- Holme P, Saramäki J (2012). “Temporal networks.” *Physics Reports*, **519**(3), 97–125. ISSN 0370-1573. doi:10.1016/j.physrep.2012.03.001. Temporal Networks.
- Keeling CD, Piper SC, Bacastow RB, Wahlen M, Whorf TP, Heimann M, Meijer HA (2005). *Atmospheric CO<sub>2</sub> and 13CO<sub>2</sub> Exchange with the Terrestrial Biosphere and Oceans from 1978 to 2000: Observations and Carbon Cycle Implications*, pp. 83–113. Springer New York, New York, NY. ISBN 978-0-387-27048-7. doi:10.1007/0-387-27048-5\_5.
- Kivelä M, Arenas A, Barthélemy M, Gleeson JP, Moreno Y, Porter MA (2014). “Multilayer networks.” *Journal of Complex Networks*, **2**(3), 203–271. ISSN 2051-1310. doi:10.1093/comnet/cnu016.
- Lacasa L, Luque B, Ballesteros F, Luque J, Nuño JC (2008). “From time series to complex networks: The visibility graph.” *Proceedings of the National Academy of Sciences*, **105**(13), 4972–4975. ISSN 0027-8424. doi:10.1073/pnas.0709247105.
- Lan X, Mo H, Chen S, Liu Q, Deng Y (2015). “Fast transformation from time series to visibility graphs.” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **25**(8), 083105. doi:10.1063/1.4927835.
- Liu C, Springer D, Li Q, Moody B, Juan RA, Chorro FJ, Castells F, Roig JM, Silva I, Johnson AEW, Syed Z, Schmidt SE, Papadaniil CD, Hadjileontiadis L, Naseri H, Moukadem A, Dieterlen A, Brandt C, Tang H, Samieinasab M, Samieinasab MR, Sameni R, Mark RG, Clifford GD (2016). “An open access database for the evaluation of heart sound algorithms.” *Physiological Measurement*, **37**(12), 2181–2213. doi:10.1088/0967-3334/37/12/2181.
- Luque B, Lacasa L, Ballesteros F, Luque J (2009). “Horizontal visibility graphs: Exact results for random time series.” *Phys. Rev. E*, **80**, 046103. doi:10.1103/PhysRevE.80.046103.
- MacKay D (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN 9780521642989.
- Meyer PE (2014). *infotheo: Information-Theoretic Measures*. R package version 1.2.0.
- Mitchell M (2006). “Complex systems: Network thinking.” *Artificial Intelligence*, **170**(18), 1194–1212. ISSN 0004-3702. doi:10.1016/j.artint.2006.10.002. Special Review Issue.



- Montero P, Vilar JA (2014). “TSclust: An R Package for Time Series Clustering.” *Journal of Statistical Software*, **62**(1), 1–43. doi:10.18637/jss.v062.i01.
- Mori U, Mendiburu A, Lozano JA (2016). “Distance Measures for Time Series in R: The TSdist Package.” *The R Journal*, **8**(2), 451–459. doi:10.32614/RJ-2016-058.
- Qiao L, Zhang L, Chen S, Shen D (2018). “Data-driven graph construction and graph learning: A review.” *Neurocomputing*, **312**, 336–351. ISSN 0925-2312. doi:10.1016/j.neucom.2018.05.084.
- Quian Quiroga R, Kreuz T, Grassberger P (2002). “Event synchronization: A simple and fast method to measure synchronicity and time delay patterns.” *Phys. Rev. E*, **66**, 041904. doi:10.1103/PhysRevE.66.041904.
- Reshef DN, Reshef YA, Finucane HK, Grossman SR, McVean G, Turnbaugh PJ, Lander ES, Mitzenmacher M, Sabeti PC (2011). “Detecting Novel Associations in Large Data Sets.” *Science*, **334**(6062), 1518–1524. doi:10.1126/science.1205438.
- Silva T, Zhao L (2018). *Machine Learning in Complex Networks*. Springer International Publishing. ISBN 9783319792347.
- Silva VF, Silva ME, Ribeiro P, Silva F (2021). “Time series analysis via network science: Concepts and algorithms.” *WIREs Data Mining and Knowledge Discovery*, **11**(3), e1404. doi:10.1002/widm.1404.
- Ting-Ting Z, Ning-De J, Zhong-Ke G, Yue-Bin L (2012). “Limited penetrable visibility graph for establishing complex network from time series.” *Acta Physica Sinica*, **61**(3), 030506. doi:10.7498/aps.61.030506.
- Tumminello M, Lillo F, Mantegna RN (2010). “Correlation, hierarchies, and networks in financial markets.” *Journal of Economic Behavior & Organization*, **75**(1), 40–58. ISSN 0167-2681. doi:10.1016/j.jebo.2010.01.004. Transdisciplinary Perspectives on Economic Complexity.
- v Rossum MCW (2001). “A Novel Spike Distance.” *Neural Computation*, **13**(4), 751–763. doi:10.1162/089976601300014321.
- Vinh NX, Epps J, Bailey J (2010). “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance.” *Journal of Machine Learning Research*, **11**(95), 2837–2854.
- Wand MP (1997). “Data-Based Choice of Histogram Bin Width.” *The American Statistician*, **51**(1), 59–64. doi:10.1080/00031305.1997.10473591.
- Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013). “Experimental comparison of representation methods and distance measures for time series data.” *Data Mining and Knowledge Discovery*, **26**(2), 275–309. doi:10.1007/s10618-012-0250-5.
- Zou Y, Donner RV, Marwan N, Donges JF, Kurths J (2019). “Complex network approaches to nonlinear time series analysis.” *Physics Reports*, **787**, 1–97. ISSN 0370-1573. doi:10.1016/j.physrep.2018.10.005. Complex network approaches to nonlinear time series analysis.

**Affiliation:**

Leonardo N. Ferreira  
Center for Humans & Machines  
Max Planck Institute for Human Development  
Lentzeallee 94, 14195 Berlin, Germany  
E-mail: [ferreira@leonardonascimento.com](mailto:ferreira@leonardonascimento.com)  
URL: <https://www.leonardoferreira.com/>