Journal of Statistical Software

November 2014, Volume 62, Issue 1. *http://www.jstatsoft.org/*

TSclust: An R Package for Time Series Clustering

Pablo Montero
University of A Coruña

José A. Vilar
University of A Coruña

Abstract

Time series clustering is an active research area with applications in a wide range of fields. One key component in cluster analysis is determining a proper dissimilarity measure between two data objects, and many criteria have been proposed in the literature to assess dissimilarity between two time series. The R package **TSclust** is aimed to implement a large set of well-established peer-reviewed time series dissimilarity measures, including measures based on raw data, extracted features, underlying parametric models, complexity levels, and forecast behaviors. Computation of these measures allows the user to perform clustering by using conventional clustering algorithms. **TSclust** also includes a clustering procedure based on p values from checking the equality of generating models, and some utilities to evaluate cluster solutions. The implemented dissimilarity functions are accessible individually for an easier extension and possible use out of the clustering context. The main features of **TSclust** are described and examples of its use are presented.

Keywords: time series data, clustering, dissimilarity measure, validation indices.

1. Introduction

Clustering is an unsupervised learning task aimed to partition a set of unlabeled data objects into homogeneous groups or clusters. Partition is performed in such a way that objects in the same cluster are more similar to each other than objects in different clusters according to some defined criterion. In many real applications, the cluster analysis must be performed on time series data. Finding stocks that behave in a similar way, determining products with similar selling patterns, identifying countries with similar population growth or regions with similar temperature are some typical applications where the similarity searching between time series is clearly motivated. In fact, the time series clustering problems arise in a natural way in a wide variety of fields, including economics, finance, medicine, ecology, environmental studies, engineering, and many others. Frequently, the grouping of series plays a central role in the studied problem. These arguments motivate the growing interest in the literature on time series clustering, especially in the last two decades where a huge number of contributions on this topic has been provided. An excellent survey on time series clustering can be seen in Liao (2005) and references therein, although significant new contributions have been made subsequently. Particularly important in the last decade has been the explosion of papers on the topic coming from both data mining and pattern recognition communities. Fu (2011) provides a complete and comprehensive overview on recent time series data mining directions, including a range of key problems such as representation, indexing and segmentation of series, measures of dissimilarity, clustering procedures and visualization tools.

A crucial question in cluster analysis is establishing what we mean by "similar" data objects, i.e., determining a suitable similarity/dissimilarity measure between two objects. In the specific context of time series data, the concept of dissimilarity is particularly complex due to the dynamic character of the series. Dissimilarities usually considered in conventional clustering could not work adequately with time dependent data because they ignore the interdependence relationship between values. This way, different approaches to define dissimilarity between time series have been proposed in the literature and a short overview of them is presented below.

If the objective is to compare profiles of series, then conventional distances between raw data (Euclidean or Manhattan, among others) evaluating a one-to-one mapping of each pair of sequences can produce satisfactory results. Sometimes, depending on the domain, the chosen dissimilarity must satisfy properties of invariance to specific distortions of the data to obtain proper results. Batista, Wang, and Keogh (2011) provide an interesting review of dissimilarity measures designed to be invariant to features like local scaling (warping), uniform scaling, offset, amplitude scaling, phase, complexity, and so on. Other times, dissimilarity is measured by comparing sequences of serial features extracted from the original time series, such as autocorrelations, cross-correlations, spectral features, wavelet coefficients, and so on (see Kovačić 1998; Struzik and Siebes 1999; Galeano and Peña 2000; Caiado, Crato, and Peña 2006; Douzal Chouakria and Nagabhushan 2007, among others). These feature-based approaches are aimed to represent the dynamic structure of each series by a feature vector of lower dimension, thus allowing a dimensionality reduction (time series are essentially high-dimensionality data) and a meaningful saving in computation time. In addition, the features commonly used in the literature for the similarity matching problem can be obtained in a straightforward manner and all of these properties result in more efficient clustering procedures. An alternative approach consists in assuming specific underlying models and evaluating dissimilarity between fitted models (some references following this approach are Piccolo 1990; Maharaj 1996, 2002; Kakizawa, Shumway, and Taniguchi 1998; Vilar and Pértega 2004, among many others). The most commonly considered criterion has been to assume that the time series are generated by ARIMA processes, although researchers in speech recognition and machine learning have also adopted alternative models as Markov chains (MC, see Ramoni, Sebastiani, and Cohen 2002) or hidden Markov models (HMM, see Smyth 1997; Oates, Firoiu, and Cohen 1999, among others). An interesting set of references on these approaches can be seen in Bagnall, Janacek, de la Iglesia, and Zhang (2003). Other dissimilarity measures are aimed to compare levels of complexity of time series (see, e.g., Li, Chen, Li, Ma, and Vitányi 2004; Sculley and Brodley 2006; Keogh, Lonardi, Ratanamahatana, Wei, Lee, and Handley 2007; Brandmaier 2011; Batista *et al.* 2011). The complexity of a time series can be estimated by following different approaches although, roughly speaking, most of them are based on the notion of Kolmogorov complexity or algorithmic entropy (Li and Vitányi 2007). Kolmogorov complexity can be thought of as the ultimate lower bound of all measures of information, but unfortunately it cannot be computed and must be approximated in practice. Two prominent approaches to evaluate complexity differences between two time series are: (i) using algorithms based on data compression (see, e.g., Li, Badger, Chen, Kwong, Kearney, and Zhang 2001; Li *et al.* 2004; Keogh, Lonardi, and Ratanamahatana 2004; Cilibrasi and Vitányi 2005; Keogh *et al.* 2007), and (ii) considering differences between permutation distributions (Brandmaier 2011). Beyond all these criteria to define dissimilarity between series, it is worth mentioning that the dissimilarity measures are often tailored for the problem at hand, highlighting properties of the time series that are of interest for the specific context. For instance, there are practical situations where the real interest of the clustering relies on the properties of the predictions, as in the case of any sustainable development problem or in situations where the concern is to reach target values on a pre-specified future time. Works by Alonso, Berrendero, Hernandez, and Justel (2006) and Vilar, Alonso, and Vilar (2010) focused on this idea and considered a notion of dissimilarity governed by the performance of future forecasts. Specifically, two time series are similar if their forecasts for a specific future time are close.

Summing up, there exist a broad range of measures to compare time series and the choice of the proper dissimilarity measure depends largely on the nature of the clustering, i.e., on determining what the purpose of the grouping is. Once the dissimilarity measure is determined, an initial pairwise dissimilarity matrix can be obtained and a conventional clustering algorithm be then used to form groups of objects. In fact, most of the time series clustering approaches reviewed by Liao (2005) are variations of general procedures (e.g., a k-means or a hierarchical clustering) that use a range of dissimilarities specifically designed to deal with time series. According to this consideration, our attention focused on the implementation of these measures, and the package **TSclust** presented in this work is the result of integrating a wide range of dissimilarity measures to perform time series clustering. The package contains commonly used dissimilarity measures, including complexity-based measures, model-based measures, feature-based measures and the prediction-based dissimilarity introduced by Vilar *et al.* (2010). Some of these measures work in the time domain but others are developed in the frequency domain, and as will be seen later, in many cases their implementation requires a range of statistical techniques (autoregressive models estimation, kernel density estimation, local polynomial regression, automatic bandwidth selectors, resampling procedures, wavelets, among others). All dissimilarity measures included in **TSclust** have been previously studied in the time series clustering literature and are properly referenced throughout this work. It is worth stressing that some dissimilarity measures work under certain regularity conditions while others are applicable in more general contexts. Indeed, **TSclust** users should carefully analyze what specific measures are more suitable to capture similarity in their clustering problem, and for it some considerations on how choosing the proper dissimilarity measure are discussed in Section 4.

Although the intended spirit of **TSclust** is not to provide a self-contained tool to perform time series clustering, some additional clustering utilities (not available in other R packages to the best of our knowledge) have been incorporated into **TSclust** as useful complements. For instance, a clustering algorithm based on checking the equality of generating models proposed by Maharaj (2000), and two clustering validation indices used by different authors in time series clustering are implemented in **TSclust** (see Section 3 for details).

In short, **TSclust** package is an attempt to integrate a wide set of dissimilarity measures between time series so that the user can compare their performance and identify useful dissimilarity criteria in a given context. Clustering is indeed a typical scenario of application of these measures, and in fact, all of them have been studied in the clustering framework. Moreover, the package was designed to provide a flexible and extensible environment, with functions accessible individually for an easier extension and use out of the clustering context. **TSclust** is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=TSclust`.

The remainder of this paper is organized as follows. The different dissimilarity measures between time series implemented in **TSclust** are introduced in Section 2. The main features of each measure are briefly discussed, and appropriate references are provided for those readers interested in a more detailed analysis of each dissimilarity measure and its properties. Some specific clustering tools included in the package and a quick overview of clustering utilities available in R are presented in Section 3. Some general considerations on the choice of a proper dissimilarity measure to perform time series clustering are given in Section 4. Section 5 is devoted to illustrate the functionality of the package via a series of applications. Specifically, the behavior in time series clustering of several dissimilarity measures is analyzed by using a set of simulated series and several illustrative applications with real data. Finally, the main conclusions are reported in Section 6.

2. Dissimilarity measures for time series in TSclust

Many dissimilarity measures between time series have been proposed in the literature and a representative set of them has been implemented in **TSclust**. This section is devoted to briefly describe the implemented measures, which have been grouped into four categories: model-free measures (Section 2.1), model-based measures (Section 2.2), complexity-based measures (Section 2.3) and prediction-based measures (Section 2.4).

In the remainder of this section, unless otherwise specified, $\boldsymbol{X}_T = (X_1, \ldots, X_T)^\top$ and $\boldsymbol{Y}_T = (Y_1, \ldots, Y_T)^\top$ denote partial realizations from two real-valued processes $X = \{X_t, t \in \mathbb{Z}\}$ and $Y = \{Y_t, t \in \mathbb{Z}\}$, respectively. Note that serial realizations of the same length T are initially assumed, although this limitation can be omitted in some cases.

2.1. Model-free approaches

A simple approach to measure the proximity between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is to consider conventional metrics based on the closeness of their values at specific points of time. Some commonly used raw-values-based dissimilarity measures are introduced below.

Minkowski distance

The Minkowski distance of order q, with q a positive integer, also called L_q-norm distance, is defined by

$$d_{L_q}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \left(\sum_{t=1}^{T} (X_t - Y_t)^q\right)^{1/q}.$$

The Minkowski distance is typically used with q being 2 (Euclidean distance) or 1 (Manhattan distance). This metric is very sensitive to signal transformations as shifting or time scaling (stretching or shrinking of time axis). On the other hand, the proximity notion relies on the closeness of the values observed at corresponding points of time so that the observations are treated as if they were independent. In particular, d_{L_q} is invariant to permutations over time.

Fréchet distance

This distance was introduced by Fréchet (1906) to measure the proximity between continuous curves, but it has been extensively used on the discrete case (see Eiter and Mannila 1994) and in the time series framework. A formal definition for the discrete case can be given as follows. Let M be the set of all possible sequences of m pairs preserving the observations order in the form

$$r = \left((X_{a_1}, Y_{b_1}), ..., (X_{a_m}, Y_{b_m}) \right),$$

with $a_i, b_j \in \{1, ..., T\}$ such that $a_1 = b_1 = 1$, $a_m = b_m = T$, and $a_{i+1} = a_i$ or $a_i + 1$ and $b_{i+1} = b_i$ or $b_i + 1$, for $i \in \{1, .., m-1\}$. Then the Fréchet distance is defined by

$$d_F\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \min_{r \in M} \left(\max_{i=1,..,m} |X_{a_i} - Y_{b_i}| \right).$$

Unlike the Minkowski distance, the Fréchet distance does not just treat the series as two points sets, but it has into account the ordering of the observations. Note that d_F can be computed on series of different length.

Dynamic time warping distance

The dynamic time warping (DTW) distance was studied in depth by Sankoff and Kruskal (1983) and proposed to find patterns in time series by Berndt and Clifford (1994). As Fréchet distance, DTW distance is aimed to find a mapping r between the series so that a specific distance measure between the coupled observations (X_{a_i}, Y_{b_i}) is minimized. The definition of the DTW distance is given by

$$d_{DTW}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \min_{r \in M} \left(\sum_{i=1,..,m} |X_{a_i} - Y_{b_i}| \right).$$

In **TSclust**, d_F and d_{DTW} are computed by using the R packages **longitudinalData** (Genolini 2014) and **dtw** (Giorgino 2009), respectively. Both distances allow to recognize similar shapes, even in the presence of signal transformations such as shifting and/or scaling. However, as in the case of d_{L_q}, both d_F and d_{DTW} ignore the temporal structure of the values as the proximity is based on the differences $|X_{a_i} - Y_{b_i}|$ regardless of the behavior around these values. A dissimilarity index model including both behavior and values proximity estimation is described below.

An adaptive dissimilarity index covering both proximity on values and on behavior

Douzal Chouakria and Nagabhushan (2007) introduce a dissimilarity measure addressed to cover both conventional measures for the proximity on observations and temporal correlation for the behavior proximity estimation. The proximity between the dynamic behaviors of the series is evaluated by means of the first order temporal correlation coefficient, which is defined by

$$CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{\sum_{t=1}^{T-1}(X_{t+1} - X_t)(Y_{t+1} - Y_t)}{\sqrt{\sum_{t=1}^{T-1}(X_{t+1} - X_t)^2}\sqrt{\sum_{t=1}^{T-1}(Y_{t+1} - Y_t)^2}}.$$

$CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ belongs to the interval $[-1, 1]$. The value $CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = 1$ means that both series show a similar dynamic behavior, i.e., their growths (positive or negative) at any instant of time are similar in direction and rate. The value $CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = -1$ implies a similar growth in rate but opposite in direction (opposite behavior). Finally, $CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = 0$ expresses that there is no monotonicity between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, and their growth rates are stochastically linearly independent (different behaviors).

The dissimilarity index proposed by Douzal Chouakria and Nagabhushan (2007) modulates the proximity between the raw-values of two series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ using the coefficient $CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$. Specifically, it is defined as follows.

$$d_{CORT}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \phi_k\left[CORT\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)\right] \cdot d\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right),$$

where $\phi_k(\cdot)$ is an adaptive tuning function to automatically modulate a conventional raw-data distance $d\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ (e.g., d_{L_q}, d_F or d_{DTW}) according to the temporal correlation. The modulating function should work increasing (decreasing) the weight of the dissimilarity between observations as the temporal correlation decreases from 0 to -1 (increases from 0 to $+1$). In addition, $d_{CORT}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ should approach the raw-data discrepancy as the temporal correlation is zero. Instead of, for instance, a linear tuning function, Douzal Chouakria and Nagabhushan propose to use an exponential adaptive function given by

$$\phi_k(u) = \frac{2}{1 + \exp\left(ku\right)}, \;\; k \geq 0.$$

Now, we focus on model-free dissimilarity measures based on particular features of the time series. The idea is to replace the raw data by a reduced number of features characterizing the time series, which allows us to take into account the temporal structure of the series.

Correlation-based distances

A first and simple dissimilarity criterion is to consider the Pearson's correlation factor between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ given by

$$COR\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{\sum_{t=1}^{T}\left(X_t - \overline{\boldsymbol{X}}_T\right)\left(Y_t - \overline{\boldsymbol{Y}}_T\right)}{\sqrt{\sum_{t=1}^{T}\left(X_T - \overline{\boldsymbol{X}}_T\right)^2}\sqrt{\sum_{t=1}^{T}\left(Y_t - \overline{\boldsymbol{Y}}_T\right)^2}},$$

with $\overline{\boldsymbol{X}}_T$ and $\overline{\boldsymbol{Y}}_T$ the average values of the serial realizations $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ respectively. Golay, Kollias, Stoll, Meier, Valavanis, and Boesiger (2005) construct a fuzzy k-means algorithm using the following two cross-correlation-based distances:

$$d_{COR.1}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{2\left(1 - COR\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)\right)},$$

and

$$d_{COR.2}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\left(\frac{1 - COR\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)}{1 + COR\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)}\right)^{\beta}}, \text{ with } \beta \geq 0.$$

Note that $d_{COR.2}$ becomes infinite when $COR\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = -1$ and the parameter β allows regulation of the fast decreasing of the distance. Graphs of functions $d_1 = d^2_{COR.1}$ and $d_2 = d^2_{COR.2}$ for some values of β are shown in Figure 1.

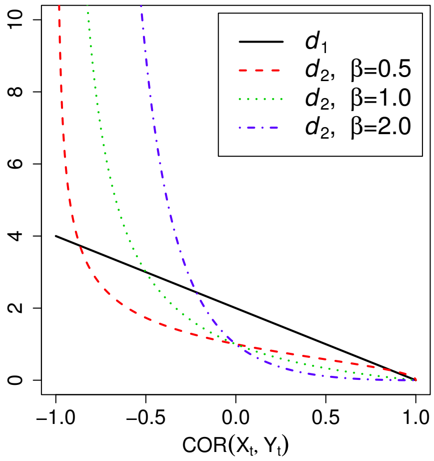

Figure 1: Plots of $d_1 = d^2_{COR.1}$ and $d_2 = d^2_{COR.2}$ as function of the cross-correlation factor for several values of β.

Autocorrelation-based distances

Several authors have considered measures based on the estimated autocorrelation functions (see e.g., Bohte, Cepar, and Košmelj 1980; Galeano and Peña 2000; Caiado *et al.* 2006; D'Urso and Maharaj 2009).

Let $\hat{\boldsymbol{\rho}}_{X_T} = (\hat{\rho}_{1,X_T}, .., \hat{\rho}_{L,X_T})^\top$ and $\hat{\boldsymbol{\rho}}_{Y_T} = (\hat{\rho}_{1,Y_T}, .., \hat{\rho}_{L,Y_T})^\top$ be the estimated autocorrelation vectors of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ respectively, for some L such that $\hat{\rho}_{i,X_T} \approx 0$ and $\hat{\rho}_{i,Y_T} \approx 0$ for $i > L$. Galeano and Peña (2000) define a distance between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ as follows.

$$d_{ACF}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\left(\hat{\boldsymbol{\rho}}_{X_T} - \hat{\boldsymbol{\rho}}_{Y_T}\right)^\top \boldsymbol{\Omega} \left(\hat{\boldsymbol{\rho}}_{X_T} - \hat{\boldsymbol{\rho}}_{Y_T}\right)},$$

where $\boldsymbol{\Omega}$ is a matrix of weights.

Some common choices of $\boldsymbol{\Omega}$ are:

(i) Consider uniform weights by taking $\boldsymbol{\Omega} = \boldsymbol{I}$. In such case d_{ACF} becomes the Euclidean distance between the estimated autocorrelation functions:

$$d_{ACFU}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\sum_{i=1}^{L} \left(\hat{\rho}_{i,X_T} - \hat{\rho}_{i,Y_T}\right)^2}.$$

(ii) Consider geometric weights decaying with the autocorrelation lag, so that d_{ACF} takes the form:

$$d_{ACFG}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\sum_{i=1}^{L} p(1-p)^i \left(\hat{\rho}_{i,X_T} - \hat{\rho}_{i,Y_T}\right)^2}, \text{ with } 0 < p < 1.$$

Analogous distances can be constructed by considering the partial autocorrelation functions (PACFs) instead of the ACFs. Hereafter, notation d_{PACFU} and d_{PACFG} will be used to denote the Euclidean distance between the estimated partial autocorrelation coefficients with uniform weights and with geometric weights decaying with the lag, respectively.

So far all metrics work in the time domain, but the frequency domain approach also offers an interesting alternative to measure the dissimilarity between time series. The key idea is to assess the dissimilarity between the corresponding spectral representations of the series.

Periodogram-based distances

Let $I_{X_T}(\lambda_k) = T^{-1}\left|\sum_{t=1}^{T} X_t\, e^{-i\lambda_k t}\right|^2$ and $I_Y(\lambda_k) = T^{-1}\left|\sum_{t=1}^{T} Y_t\, e^{-i\lambda_k t}\right|^2$ be the periodograms of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, at frequencies $\lambda_k = 2\pi k/T$, $k = 1, \ldots, n$, with $n = [(T-1)/2]$. Three dissimilarity measures based on periodograms were analyzed by Caiado *et al.* (2006).

(i) The Euclidean distance between the periodogram ordinates:

$$d_P\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{1}{n}\sqrt{\sum_{k=1}^{n}\left(I_{X_T}(\lambda_k) - I_{Y_T}(\lambda_k)\right)^2}.$$

(ii) If we are not interested in the process scale but only on its correlation structure, better results can be obtained using the Euclidean distance between the normalized periodogram ordinates:

$$d_{NP}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{1}{n}\sqrt{\sum_{k=1}^{n}\left(NI_{X_T}(\lambda_k) - NI_{Y_T}(\lambda_k)\right)^2},$$

where $NI_{X_T}(\lambda_k) = I_{X_T}(\lambda_k)/\widehat{\gamma}_{0,X_T}$ and $NI_{Y_T}(\lambda_k) = I_{Y_T}(\lambda_k)/\widehat{\gamma}_{0,Y_T}$ with $\widehat{\gamma}_{0,X_T}$ and $\widehat{\gamma}_{0,Y_T}$ being the sample variances of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively.

(iii) As the variance of the periodogram ordinates is proportional to the spectrum value at the corresponding frequencies, it makes sense to use the logarithm of the normalized periodogram:

$$d_{LNP}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{1}{n}\sqrt{\sum_{k=1}^{n}\left(\log NI_{X_T}(\lambda_k) - \log NI_{Y_T}(\lambda_k)\right)^2}.$$

Casado de Lucas (2010) considers a distance measure based on the cumulative versions of the periodograms, i.e., the integrated periodograms. Casado de Lucas argues that the approaches based on the integrated periodogram present several advantages over the ones based on periodograms. In particular,

- The periodogram is an asymptotically unbiased but inconsistent estimator of the spectral density while the integrated periodogram is a consistent estimator of the spectral distribution.

- From a theoretical point of view, the spectral distribution always exists, but the spectral density exists only under absolutely continuous distributions.

- The integrated periodogram completely determines the stochastic process.

The following distances based on the integrated periodogram, one normalized and other non-normalized, are proposed in Casado de Lucas (2010).

$$d_{IP}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \int_{-\pi}^{\pi}\left|F_{X_T}(\lambda) - F_{Y_T}(\lambda)\right| d\lambda,$$

where $F_{X_T}(\lambda_j) = C_{X_T}^{-1} \sum_{i=1}^{j} I_{X_T}(\lambda_i)$ and $F_{Y_T}(\lambda_j) = C_{Y_T}^{-1} \sum_{i=1}^{j} I_{Y_T}(\lambda_i)$, with $C_{X_T} = \sum_i I_X(\lambda_i)$ and $C_{Y_T} = \sum_i I_{Y_T}(\lambda_i)$ for the normalized version, and $C_{X_T} = C_{Y_T} = 1$ for the non-normalized version.

The normalized version gives more weight to the shape of the curves while the non-normalized considers the scale. Casado de Lucas suggests using the normalized version when the graphs of the functions tend to intersect, and the non-normalized when they do not.

Dissimilarity measures based on nonparametric spectral estimators

Kakizawa *et al.* (1998) proposed a general spectral disparity measure between two series given by

$$d_W\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{1}{4\pi} \int_{-\pi}^{\pi} W\left(\frac{f_{X_T}(\lambda)}{f_{Y_T}(\lambda)}\right) d\lambda, \qquad (1)$$

where f_{X_T} and f_{Y_T} denote the spectral densities of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, and $W(\cdot)$ is a divergence function satisfying appropriate regular conditions to ensure that d_W has the quasi-distance property. If, for example, $W(x) = \log(\alpha x + (1-\alpha)) - \alpha \log x$, with $0 < \alpha < 1$, then d_W corresponds to the limiting spectral approximation of the Chernoff information in the time domain (Shumway and Unger 1974). Note that d_W is not a real distance because it is not symmetric and does not satisfy the triangle inequality. For clustering, it is more convenient to modify the divergence function by setting $\tilde{W}(x) = W(x) + W(x^{-1})$.

In practice, the spectra f_{X_T} and f_{Y_T} are unknown and must be previously estimated. Vilar and Pértega (2004) studied the asymptotic properties of d_W when f_{X_T} and f_{Y_T} are replaced by nonparametric estimators constructed via local linear regression. These approximations can be done in three different ways (Fan and Kreutzberger 1998), thus resulting three different versions of the d_W dissimilarity measure. Specifically,

- $d_{W(DLS)}$, when the spectra are replaced by local lineal smoothers of the periodograms, obtained via least squares.
- $d_{W(LS)}$, when the spectra are replaced by the exponential transformation of local linear smoothers of the log-periodograms, obtained via least squares.
- $d_{W(LK)}$, when the spectra are replaced by the exponential transformation of local linear smoothers of the log-periodograms, here obtained by using the maximum local likelihood criterion.

Due to the asymptotic inefficiency of $d_{W(LS)}$ with respect to both $d_{W(DLS)}$ and $d_{W(LK)}$, only these latter two metric are implemented in **TSclust**. In particular, the weighted least squares smoother is obtained using the R package **locpol** (Ojeda Cabrera 2012). The default value of the bandwidth is an automatic plug-in selector specifically designed for local linear Gaussian kernel regression (see Ruppert, Sheather, and Wand 1995). This plug-in method is implemented using the R package **KernSmooth** (Wand 2014).

Two alternative nonparametric spectral dissimilarity measures introduced by Pértega and Vilar (2010) are also implemented in **TSclust**. In both cases, the discrepancy measure is given by a nonparametric statistic originally introduced to check the equality of the log-spectra of two processes.

The first alternative comes from the generalized likelihood ratio test approach introduced by Fan and Zhang (2004) to check whether the density of an observed time series belongs to a parametric family. Pértega and Vilar (2010) introduce a slight modification of this test statistic in order to check the equality of two log-spectra, resulting

$$d_{GLK}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sum_{k=1}^{n}\left[Z_k - \hat{\mu}(\lambda_k) - 2\log(1 + e^{\{Z_k - \hat{\mu}(\lambda_k)\}})\right] - \sum_{k=1}^{n}\left[Z_k - 2\log(1 + e^{Z_k})\right],$$

where $Z_k = \log(I_{X_T}(\lambda_k)) - \log(I_{Y_T}(\lambda_k))$, and $\hat{\mu}(\lambda_k)$ is the local maximum log-likelihood estimator of $\mu(\lambda_k) = \log(f_{X_T}(\lambda_k)) - \log(f_{Y_T}(\lambda_k))$ computed by local linear fitting.

The second distance evaluates the integrated squared differences between nonparametric estimators of the log-spectra and it is given by

$$d_{ISD}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \int_{-\pi}^{\pi}\left(\hat{m}_{X_T}(\lambda) - \hat{m}_{Y_T}(\lambda)\right)^2 d\lambda,$$

where $\hat{m}_{X_T}(\lambda)$ and $\hat{m}_{Y_T}(\lambda)$ are local linear smoothers of the log-periodograms obtained using the maximum local likelihood criterion.

In all cases, local linear smoothing techniques were considered to approximate the unknown log-spectra because of their good theoretical and practical properties, such as nice minimax efficiency properties and absence of boundary effects, among others (Fan and Gijbels 1996). Moreover, these properties are satisfied under very general regularity conditions of the spectral densities, thus providing a great versatility to approximate spectra from different kinds of processes. Hence, it is intuitively expected that these methods are more robust in cluster analysis than other approaches based on *ad hoc* parametric modeling.

A dissimilarity measure based on the discrete wavelet transform

Discrete wavelet transform (DWT) is a useful feature extraction technique often used to measure dissimilarity between time series. DWT performs a scale-wise decomposing of the time series in such a way that most of the energy of the time series can be represented by only a few coefficients. The basic idea is to replace the original series by their wavelet approximation coefficients in an appropriate scale, and then to measure the dissimilarity between the wavelet approximations. A detailed description of wavelet methods for time series analysis can be seen in Percival and Walden (2006) and some interesting references where wavelets are used to clustering time series are Struzik and Siebes (1999), Chan and Fu (1999), Popivanov and Miller (2002), Chan, Fu, and Yu (2003) and Zhang, Ho, Zhang, and Lin (2006), among others.

There is indeed a key point when we wish to use the DWT technique for clustering: the choice of an appropriate scale to obtain an accurate clustering. In **TSclust**, an automatic algorithm to determine this scale is implemented. The algorithm was proposed by Zhang *et al.* (2006) and it is aimed to select the scale by levering two conflicting requirements: an efficient reduction of the dimensionality and preserving as much information from the original data as possible. Specifically, the algorithm works as follows.

Consider a set of time series $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(m)}\right\}$ located in a scale $J = \log_2(T)$. Denote by $\boldsymbol{H}_j\left(\boldsymbol{X}_T^{(i)}\right) = \left\{\boldsymbol{A}_j^{(i)}, \boldsymbol{D}_j^{(i)}, \boldsymbol{D}_{j+1}^{(i)}, \ldots, \boldsymbol{D}_{J-1}^{(i)}\right\}$ the coefficients corresponding to the discrete wavelet transform of $\boldsymbol{X}_T^{(i)}$ at the scale j. The $\boldsymbol{A}_j^{(i)}$ are called approximation coefficients (or smooth coefficients) and represent the smooth behavior of the data. The $\boldsymbol{D}_k^{(i)}$, $k = j, j+1, \ldots, J-1$, are called the detail coefficients because they represent the finer, more high-frequency nature, of the data. As feature vector, Zhang *et al.* propose to retain the approximation coefficients $\boldsymbol{A}_j^{(i)}$ for a particular scale $j^\star$ corresponding to the highest scale that satisfies:

$$\sum_{i=1}^{m} E\left(\boldsymbol{D}_{j^\star}^{(i)}\right) < \sum_{i=1}^{m} E\left(\boldsymbol{D}_{j^\star-1}^{(i)}\right), \tag{2}$$

where $E\left(\boldsymbol{Z}\right) = \sum_{k=1}^{s} z_k^2$ denotes the energy associated with a vector $\boldsymbol{Z} \in \mathbb{R}^s$. Argument behind this criterion is that the sum of squared errors between $\boldsymbol{X}_T^{(i)}$ and the reconstructed approximation series $\hat{\boldsymbol{X}}_T^{(i)}$ is given by

$$SSE\left(\boldsymbol{X}_T^{(i)}, \hat{\boldsymbol{X}}_T^{(i)}\right) = E\left(\boldsymbol{X}_T^{(i)}\right) - E\left(\boldsymbol{A}_j^{(i)}\right) = \sum_{k=j}^{J-1} E\left(\boldsymbol{D}_k^{(i)}\right).$$

Therefore, removing the detail coefficients within a scale $j^\star$ satisfying Equation 2 and higher scales means to achieve a tradeoff between lower dimensionality and lower $SSE\left(\boldsymbol{X}_T^{(i)}, \hat{\boldsymbol{X}}_T^{(i)}\right)$ (i.e., lower energy loss). Moreover, Zhang *et al.* show that the proposed algorithm is efficient when using Haar wavelet. Once the appropriate scale $j^\star$ is determined, the dissimilarity between two series $\boldsymbol{X}_T^{(u)}$ and $\boldsymbol{X}_T^{(v)}$, with $u, v \in \{1, \ldots, m\}$, $u \neq v$, is given by

$$d_{DWT}\left(\boldsymbol{X}_T^{(u)}, \boldsymbol{X}_T^{(v)}\right) = \sqrt{\sum_k \left(a_{k,j^\star}^{(u)} - a_{k,j^\star}^{(v)}\right)^2},$$

where $a_{k,j^\star}^{(u)}$ and $a_{k,j^\star}^{(v)}$ are the elements of $\boldsymbol{A}_{j^\star}^{(u)}$ and $\boldsymbol{A}_{j^\star}^{(v)}$, respectively.

In **TSclust**, the discrete wavelet transform of the time series is computed using the R package **wmtsa** (Constantine and Percival 2014).

Dissimilarity based on the symbolic representation SAX

Symbolization involves transformation of time series into sequences of discretized symbols that can be efficiently processed to extract information about the underlying processes. Although different techniques to generate symbolic representations of time series have been introduced in the literature (Daw, Finney, and Tracy 2003), many of them fail to define a dissimilarity measure in the symbolic space due to several reasons, such as producing a poor dimension reduction or showing little correlation with a dissimilarity measure in the original space. The symbolic representation SAX (symbolic aggregate approximation) introduced by Lin, Keogh, Lonardi, and Chiu (2003) does not suffer from these flaws. The SAX approach first transforms the original data into the piecewise aggregate approximation (PAA) representation (Yi and Faloutsos 2000; Keogh, Chakrabarti, Pazzani, and Mehrotra 2000) and then symbolizes the PAA representation into a discrete string. The PAA intermediate representation guarantees a high dimensionality reduction power, and furthermore, allows to prove that a dissimilarity based on the symbolic strings lower bounds the true dissimilarity between the original time series (Lin *et al.* 2003). As result, SAX representation is highly competitive to deal with a variety of data mining problems, including indeed cluster analysis of time series. The SAX approach is outlined below.

1. **Normalization** Time series are transformed to have zero mean and unit variance.

2. **PAA representation** Each normalized series $\boldsymbol{X}_T$ is represented in a w-dimensional space by the vector $\overline{\boldsymbol{X}}_w = \left(\overline{X}_1, \ldots, \overline{X}_w\right)$, whose i-th element is calculated by

$$\overline{X}_i = \frac{w}{T} \sum_{j=\frac{T}{w}(i-1)+1}^{\frac{T}{w}i} X_j. \tag{3}$$

   Thus, data are first divided into w segments of equal-length. Then, the i-th component of the PAA representation is given by the mean value of the data falling within the i-th segment. Equation 3 assumes that T/ω is an integer, but this assumption can be relaxed by weighting the contribution of data points placed in adjacent segments (see details on this generalization in Section 3.5 of Lin, Keogh, Wei, and Lonardi 2007).

3. **SAX representation** Once the number α of symbols (letters of the alphabet) is selected, the set of α^{-1}-quantiles of the $N(0,1)$ density, $\{z_{1/\alpha}, z_{2/\alpha}, \ldots, z_{(\alpha-1)/\alpha}\}$, is computed to determine equal-sized zones under Gaussian curve. Now, if l_i denotes the i-th letter of the alphabet, for $i = 1, \ldots, \alpha$, then each PAA representation $\overline{\boldsymbol{X}}_w$ is mapped to a concatenation of α symbols (so-called "word"), let say $\hat{\boldsymbol{X}}_\alpha = \left(\hat{X}_1, \ldots, \hat{X}_\alpha\right)$, as follows.

$$\hat{X}_i = \begin{cases} l_1 & \text{if } \overline{X}_i < z_{1/\alpha} \\ l_j & \text{if } \overline{X}_i \in \left[z_{(j-1)/\alpha}\ , \ z_{j/\alpha}\right) \\ l_\alpha & \text{if } \overline{X}_i \geq z_{(\alpha-1)/\alpha} \end{cases}$$

   This way, each symbol of the resulting word $\hat{\boldsymbol{X}}_\alpha$, i.e., of the SAX symbolic representation, is generated with approximately the same probability.

4. **MINDIST dissimilarity measure** Though many dissimilarities can be defined over the SAX representation, one that approximates the Euclidean distance can be useful. First, the distance between a pair of symbols l_i and l_j, $i, j \in \{1, \ldots, \alpha\}$, is defined by

$$d_\alpha\left(l_i, l_j\right) = \begin{cases} 0 & \text{if } |i-j| \leq 1 \\ z_{(\max(i,j)-1)/\alpha} - z_{\min(i,j)/\alpha} & \text{otherwise} \end{cases}$$

   Note that the square matrix containing all pairwise distances between symbols needs only be computed once. Based on this matrix, the dissimilarity is directly calculated as follows

$$d_{MINDIST.SAX}\left(\hat{\boldsymbol{X}}_\alpha, \hat{\boldsymbol{Y}}_\alpha\right) = \sqrt{\frac{T}{w}} \sqrt{\sum_{i=1}^{w} \left[d_\alpha\left(\hat{X}_i, \hat{Y}_i\right)\right]^2}.$$

A complete information on the SAX approach and related issues can be seen at the web page Keogh (2014) and references therein. **TSclust** incorporates routines to generate the PAA and SAX representations and compute the dissimilarity $d_{MINDIST.SAX}$.

2.2. Model-based approaches

Model-based dissimilarity measures assume that the underlying models are generated from specific parametric structures. The main approach in the literature is to assume that the generating processes of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ follow invertible ARIMA models. In such case, the idea is fitting an ARIMA model to each series and then measuring the dissimilarity between the fitted models. First step requires estimating the structure and the parameters of ARIMA models. The structure is either assumed to be given or automatically estimated using, for example, the Akaike's information criterion (AIC) or the Schawartz's Bayesian information criterion (BIC). The parameter values are commonly fitted using generalized least squares estimators. Some of the most relevant dissimilarity measures derived in the literature under the assumption of underlying ARIMA models are provided below.

Piccolo distance

Piccolo (1990) defines a dissimilarity measure in the class of invertible ARIMA processes as the Euclidean distance between the AR(∞) operators approximating the corresponding ARIMA structures. Piccolo argues that the autoregressive expansions convey all the useful information about the stochastic structure of this kind of processes (except for initial values). If the series are non-stationary, differencing is carried out to make them stationary, and if the series possess seasonality, then it should be removed before further analysis. Then, a definite criterion such as AIC or BIC is used to fit truncated AR(∞) models of orders k_1 and k_2 that approximate the generating processes of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively. This approach allows us to overcome the problem of obtaining *ad hoc* ARMA approximations for each of the series subjected to clustering.

If $\hat{\boldsymbol{\Pi}}_{X_T} = (\hat{\pi}_{1,X_T}, \ldots, \hat{\pi}_{k_1,X_T})^\top$ and $\hat{\boldsymbol{\Pi}}_{Y_T} = (\hat{\pi}_{1,Y_T}, \ldots, \hat{\pi}_{k_2,Y_T})^\top$ denote the vectors of AR(k_1) and AR(k_2) parameter estimations for $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, then the Piccolo's distance takes the form

$$d_{PIC}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\sum_{j=1}^{k} \left(\hat{\pi}'_{j,X_T} - \hat{\pi}'_{j,Y_T}\right)^2},$$

where $k = \max(k_1, k_2)$, $\hat{\pi}'_{j,X_T} = \hat{\pi}_{j,X_T}$, if $j \leq k_1$, and $\hat{\pi}'_{j,X_T} = 0$ otherwise, and analogously $\hat{\pi}'_{j,Y_T} = \hat{\pi}_{j,Y_T}$, if $j \leq k_2$, and $\hat{\pi}'_{j,Y_T} = 0$ otherwise.

Besides satisfying the properties of a distance (non-negativity, symmetry and triangularity), d_{PIC} always exists for any invertible ARIMA process since $\sum \pi_j$, $\sum \|\pi_j\|$ and $\sum \pi_j^2$ are well-defined quantities.

Maharaj distance

For the class of invertible and stationary ARMA processes, Maharaj (1996, 2000) introduced two discrepancy measures based on hypotheses testing to determine whether or not two time series have significantly different generating processes. The first of these metrics is given by the test statistic

$$d_{MAH}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{T}\left(\hat{\boldsymbol{\Pi}}'_{X_T} - \hat{\boldsymbol{\Pi}}'_{Y_T}\right)^\top \hat{\boldsymbol{V}}^{-1}\left(\hat{\boldsymbol{\Pi}}'_{X_T} - \hat{\boldsymbol{\Pi}}'_{Y_T}\right),$$

where $\hat{\boldsymbol{\Pi}}'_{X_T}$ and $\hat{\boldsymbol{\Pi}}'_{Y_T}$ are the AR(k) parameter estimations of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, with k selected as in the Piccolo's distance, and $\hat{\boldsymbol{V}}$ is an estimator of $\boldsymbol{V} = \sigma^2_{X_T}\boldsymbol{R}^{-1}_{X_T}(k) + \sigma^2_{Y_T}\boldsymbol{R}^{-1}_{Y_T}(k)$, with $\sigma^2_{X_T}$ and $\sigma^2_{Y_T}$ denoting the variances of the white noise processes associated with $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, and $\boldsymbol{R}_{X_T}$ and $\boldsymbol{R}_{Y_T}$ the sample covariance matrices of both series.

Maharaj demonstrated that d_{MAH} is asymptotically χ^2 distributed under the null hypothesis of equality of generating processes, i.e., by assuming that $\mathbf{\Pi}_{X_T} = \mathbf{\Pi}_{Y_T}$. Therefore, the dissimilarity between $\hat{\mathbf{\Pi}}'_{X_T}$ and $\hat{\mathbf{\Pi}}'_{Y_T}$ can also be measured through the associated p value, i.e., by considering

$$d_{MAH,p}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = P\left(\chi^2_k > d_{MAH}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)\right).$$

Both the test statistic d_{MAH} and the associated p value $d_{MAH,p}$ satisfy the properties of non-negativity and symmetry so that any of them can be used as dissimilarity measure between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$. Although d_{MAH} and d_{PIC} evaluate the dissimilarity between two series by comparing their autoregressive approximations, there is a substantial difference between them: the Piccolo's distance does not take into account the variance of the white noise processes associated with the observed series, while the Maharaj's statistic involves these variances in its definition. It is important to be aware of this fact when we use these dissimilarity measures to carry out clustering because d_{MAH} will be affected by the scale unit.

It is also worth emphasizing that if an hierarchical algorithm starting from the pairwise matrix of p values $d_{MAH,p}$ is developed, then a clustering homogeneity criterion is implicitly provided by pre-specifying a threshold significance level α (e.g., 5% or 1%). Those series with associated p values greater than α will be grouped together, which implies that only those series whose dynamic structures are not significantly different at level α will be placed in the same group.

Measures d_{MAH} and $d_{MAH,p}$ come from a hypothesis testing procedure designed to compare two independent time series. To overcome this limitation, Maharaj (2000) introduced a new testing procedure that can be applied to time series that are not necessarily independent. In this case, a pooled model including collectively the models fitted to $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is considered, and the combined vector of $2k$ AR parameters $\mathbf{\Pi} = (\mathbf{\Pi}_{X_T}, \mathbf{\Pi}_{Y_T})^\top$ is estimated by using generalized least squares. Assuming that the two models are correlated at the same points in time but uncorrelated across observations, the proposed test statistic (say d_{MAHext}) is also asymptotically distributed as χ^2 with k degrees of freedom. As before, a dissimilarity measure (say $d_{MAHext,p}$) based on the p values associated with this new test can be constructed.

Cepstral-based distance

Kalpakis, Gada, and Puttagunta (2001) propose the use of the linear predictive coding (LPC) cepstrum for clustering ARIMA time series. The cepstrum is defined as the inverse Fourier transform of the short-time logarithmic amplitude spectrum. The cepstrum constructed by using the autoregression coefficients from linear model of the signal is referred to as the LPC Cepstrum, since it is derived from the linear predictive coding of the signal. Kalpakis *et al.* argue that LPC cepstral coefficients present good properties to discriminate between ARIMA time series. In particular, only a few LPC cepstral coefficients retains high amount of information on the underlying ARIMA model.

Consider a time series $\boldsymbol{X}_T$ following an AR(p) structure, i.e., $X_t = \sum_{r=1}^{p} \phi_r X_{t-r} + \varepsilon_t$, where ϕ_r are the autoregression coefficients and ε_t is a white noise process with zero mean and non-zero variance. Then the LPC cepstral coefficients can be derived from the autoregressive coefficients ϕ_r as follows:

$$\psi_h = \begin{cases} \phi_1 & \text{if } h = 1 \\ \phi_h + \sum_{m=1}^{h-1}(\phi_m - \psi_{h-m}) & \text{if } 1 < h \leq p \\ \sum_{m=1}^{p}(1 - \frac{m}{h})\phi_m \psi_{h-m} & \text{if } p < h \end{cases}$$

In order to measure the distance between two time series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, Kalpakis *et al.* (2001) consider the Euclidean distance between their corresponding estimated LPC cepstral coefficients, taking the form

$$d_{LPC.Cep}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \sqrt{\sum_{i=1}^{T} \left(\psi_{i,X_T} - \psi_{1,Y_T}\right)^2}.$$

2.3. Complexity-based approaches

A representative group of dissimilarity measures based on comparing levels of complexity of time series are presented in this section. Here, similarity of two time series does not rely on specific serial features or the knowledge of underlying models, but on measuring the level of shared information by both time series. The mutual information between two series can be formally established using the Kolmogorov complexity concept, although this measure cannot be computed in practice and must be approximated. A pair of approaches to measure complexity differences between two time series are shortly described below.

Compression-based dissimilarity measures

The Kolmogorov complexity $K(x)$ of an object x is the length of the shortest program capable to produce x on a universal computer, such as a Turing machine (Li and Vitányi 2007). Intuitively, $K(x)$ is the minimal quantity of information required to generate x by an algorithm, and therefore, the level of complexity of x is related to $K(x)$. Analogously, given two objects x and y, the conditional Kolmogorov complexity $K(x|y)$ of x given y is defined as the length of the shortest program producing x when y is given as an auxiliary input on the program. Therefore, $K(x) - K(x|y)$ measures the amount of information that y produces about x. Based on these concepts, Li *et al.* (2004) propose a normalized information distance (NID) between two objects x and y given by

$$d_{NID}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}.$$

Li *et al.* show that d_{NID} is a metric taking values in $[0,1]$ and it is universal in the following sense: d_{NID} leads to the smallest values (up to constant precision) among a broad class of normalized distances. Metric d_{NID} can be applied to different collections of objects such as time series, images, texts, etc. From now on we assume that x and y represent times series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively.

As Kolmogorov complexity is noncomputable, d_{NID} is approximated by replacing the quantities $K(\cdot)$ by the length of the compressed objects obtained from data compressors such as gzip, bzip2, etc. Consider a specific data compression algorithm and denote by $C(\boldsymbol{X}_T)$ the compressed size of $\boldsymbol{X}_T$. The denominator of d_{NID} is easy to approximate by $\max\left\{C(\boldsymbol{X}_T), C(\boldsymbol{Y}_T)\right\}$, but the numerator involves conditional Kolmogorov complexities making more difficult to obtain the approximation. Li *et al.* (2004) overcome this drawback by taking into account that $K(x|y)$ is roughly equal to $K(xy) - K(y)$, where $K(xy)$ is the length of the shortest program to compute the concatenation of x and y. The resulting approximation by following this approach is called "normalized compression distance" (NCD) and takes the form

$$d_{NCD}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{C\left(\boldsymbol{X}_T\boldsymbol{Y}_T\right) - \min\left\{C(\boldsymbol{X}_T), C(\boldsymbol{Y}_T)\right\}}{\max\left\{C(\boldsymbol{X}_T), C(\boldsymbol{Y}_T)\right\}}.$$

The NCD dissimilarity takes nonnegative values ranging from 0 to $1+\varepsilon$, where ε is due to flaws in the compression techniques. The smaller the $d_{NCD}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$, the more closely related $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ are. The optimality properties satisfied by the theoretical version d_{NID} do not directly hold for d_{NCD}, although Cilibrasi and Vitányi (2005) show that d_{NCD} satisfies the metric properties and approximates optimality when a specific notion of "normal" compressor is considered. The axiomatic notion of normal compressor includes symmetry as one of the basic features, and the authors argue that all standard compressors are asymptotically normal. The good performance of d_{NCD} has been illustrated in Cilibrasi and Vitányi (2005) on a wide set of experiments in areas as diverse as genomics, virology, languages, literature, music, handwritten digits and astronomy. Moreover, the dissimilarity turns out to be robust under change of the underlying compressor-types: statistical (PPMZ), Lempel-Ziv based dictionary (gzip) or block based (bzip2).

Keogh *et al.* (2004) (see also Keogh *et al.* 2007) use a simplified version of the NCD dissimilarity called "compression-based dissimilarity measure" (CDM) defined as

$$d_{CDM}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{C\left(\boldsymbol{X}_T \boldsymbol{Y}_T\right)}{C(\boldsymbol{X}_T) C(\boldsymbol{Y}_T)}.$$

The CDM dissimilarity ranges from 1/2 to 1, where 1/2 shows pure identity and 1 shows maximum discrepancy. Although Keogh *et al.* (2004) do not provide a theoretical analysis (d_{CDM} is not a metric), they emphasize that their simpler proposal produces successful results in clustering and anomaly detection.

Other compression-based dissimilarity measures have been proposed in the literature. In particular, four different compression-based measures, including NCD and CDM, are compared in an experimental study carried out by Sculley and Brodley (2006), concluding that the four examined methods lead to very similar results. In **TSclust**, d_{NCD} and d_{CDM} are implemented using the compression function `memCompress()` included in the **base** package of R.

Permutation distribution clustering

Permutation distribution clustering (PDC) represents an alternative complexity-based approach to clustering time series. Dissimilarity between series is described in terms of divergence between permutation distributions of order patterns in m-embedding of the original series. Specifically, given $\boldsymbol{X}_T$, an m-dimensional embedding is constructed by considering

$$\boldsymbol{\mathcal{X}}'_m \equiv \left\{ \boldsymbol{X}'_m = \left(X_t, X_{t+1}, \ldots, X_{t+m}\right), t = 1, \ldots, T-m \right\}.$$

Then, for each $\boldsymbol{X}'_m \in \boldsymbol{\mathcal{X}}'_m$, permutation $\Pi\left(\boldsymbol{X}'_m\right)$ obtained by sorting $\boldsymbol{X}'_m$ in ascending order (so-called *codeword of* $\boldsymbol{X}'_m$) is recorded, and the distribution of these permutations on $\boldsymbol{\mathcal{X}}'_m$, $P\left(\boldsymbol{X}_T\right)$ (so-called *codebook of* $\boldsymbol{X}_T$), is used to characterize the complexity of $\boldsymbol{X}_T$. Furthermore, dissimilarity between two time series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is measured in terms of the dissimilarity between their codebooks $P\left(\boldsymbol{X}_T\right)$ and $P\left(\boldsymbol{Y}_T\right)$, respectively. Brandmaier (2011) establishes this dissimilarity as the α-divergence between codebooks. The α-divergence concept (Amari 2007) generalizes the Kullback-Leibler divergence and the parameter α can be chosen to obtain a symmetric divergence. An interesting discussion on the use of different divergence measures and the nice properties of the permutation distribution to perform clustering (computational efficiency, robustness to drift, invariance to differences in mean and variance, among others) can be seen in Brandmaier (2011).

The choice of the embedding dimension m is the crucial point in PDC approach. A small value of m might lead to a permutation distribution with a low representational power, while a large value of m quickly leads to less reliable estimates of codeword frequencies. Brandmaier (2011) proposes a heuristic procedure to automatically select the embedding dimension, thus making a parameter-free clustering method.

A package specifically devoted to this approach, **pdc** (Brandmaier 2014), is available on the CRAN package repository. For it, the dissimilarity based on permutation distributions has not been implemented in our package and it is directly computed by using the corresponding function in **pdc**, which is automatically loaded when **TSclust** is installed.

A complexity-invariant dissimilarity measure

Batista *et al.* (2011) argue that, under many dissimilarity measures, pairs of time series with high levels of complexity frequently tend to be further apart than pairs of simple series. This way, complex series are incorrectly assigned to classes with less complexity. In order to mitigate this effect, the authors propose to use information about complexity difference between two series as a correction factor for existing dissimilarity measures. Specifically, a general complexity-invariant dissimilarity measure called CID is defined as

$$d_{CID}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = CF\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) \cdot d\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right),$$

where $d\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ denotes a conventional raw-data distance and $CF\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ is a complexity correction factor given by

$$CF\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \frac{\max\left\{CE\left(\boldsymbol{X}_T\right), CE\left(\boldsymbol{Y}_T\right)\right\}}{\min\left\{CE\left(\boldsymbol{X}_T\right), CE\left(\boldsymbol{Y}_T\right)\right\}},$$

with $CE\left(\boldsymbol{X}_T\right)$ a complexity estimator of $\boldsymbol{X}_T$. If all series have the same complexity, then $d_{CID}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = d\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$. Nevertheless, an important complexity difference between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ turns into an increase of the dissimilarity between them. The complexity estimator considered by Batista *et al.* is very simple and consists in computing

$$CE\left(\boldsymbol{X}_T\right) = \sqrt{\sum_{t=1}^{T-1}(X_t - X_{t+1})^2}.$$

The CID method is intuitive, parameter-free, invariant to the complexity of time series, computationally efficient, and it has produced improvements in accuracy in several clustering experiments carried out in Batista *et al.* (2011). In **TSclust**, d_{CID} has been implemented by using the Euclidean distance as dissimilarity measure d.

2.4. Prediction-based approaches

Now we focus on a new dissimilarity notion governed by the performance of future forecasts, i.e., two time series are similar if their forecasts at a specific future time are close. Obviously, a clustering procedure based on this dissimilarity concept may produce results very different to the ones generated from model-based or feature-based clustering methods. For instance, two time series coming from the same generating process can produce different forecasts at a pre-specified horizon, and hence these series might not be clustered together by using this new dissimilarity criterion. There are many practical situations where the real interest of the clustering relies directly on the properties of the predictions, as in the case of any sustainable development problem or in situations where the concern is to reach target values on pre-specified future time periods. Alonso *et al.* (2006) proposed a dissimilarity measure based on comparing the forecast densities for each series at a future horizon of interest. They argue that using full forecast densities permits to take into account the variability of the predictions, which is completely ignored when comparisons are based on pointwise forecasts. In practice, the forecast densities are unknown and must be approximated from the data. Alonso *et al.* construct this approximation using a smoothed sieve bootstrap procedure combined with kernel density estimation techniques. This procedure requires assuming that the time series admit an AR(1) representation because the sieve bootstrap is based on resampling residuals from autoregressive approximations. Vilar *et al.* (2010) extend this methodology to cover the case of nonparametric models of arbitrary autoregressions. In this new scenario, the sieve bootstrap is not valid, and the forecast densities are approximated considering a bootstrap procedure that mimics the generating processes without assuming any parametric model for the true autoregressive structure of the series. The most general procedure proposed by Vilar *et al.* (2010) has been implemented in **TSclust**, allowing thus the classification of general autoregressive models, including extensively studied parametric models, such as the threshold autoregressive (TAR), the exponential autoregressive (EXPAR), the smooth-transition autoregressive (STAR) and the bilinear, among others (see Tong and Yeung 2000, and references therein).

Specifically, let $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ be realizations of stationary processes that admit a general autoregressive representation of the form $S_t = \varphi(S_{t-1}) + \varepsilon_t$, with $\{\varepsilon_t\}$ an i.i.d. sequence and $\varphi(\cdot)$ a smooth function not restricted to any pre-specified parametric model. Given a particular future time $T+h$, Vilar *et al.* introduce the following distance between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$:

$$d_{PRED,h}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right) = \int \left| \hat{f}_{X_{T+h}}(u) - \hat{f}_{Y_{T+h}}(u) \right| du, \tag{4}$$

where $\hat{f}_{X_{T+h}}$ and $\hat{f}_{Y_{T+h}}$ denote estimates of the forecast densities at horizon $T+h$ for $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively.

By construction, $d_{PRED,h}\left(\boldsymbol{X}_T, \boldsymbol{Y}_T\right)$ measures the distance between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ in terms of the disparity between the behaviors of their predicted values at horizon $T+h$. The true forecast densities are replaced by kernel-type estimators based on bootstrap predictions. Although different resampling procedures can be used to obtain the bootstrap predictions, we consider a bootstrap procedure based on generating a process

$$S_t^* = \hat{\varphi}_g(S_{t-1}^*) + \varepsilon_t^*,$$

where $\hat{\varphi}_g$ is a nonparametric estimator of φ and $\{\varepsilon_t^*\}$ is a conditionally i.i.d. resample from the nonparametric residuals. This bootstrap method, called *autoregression bootstrap*, completely mimics the dependence structure of the underlying process. Actually autoregression bootstrap uses an approach similar to that of the residual-based resampling of linear autoregressions, but it takes advantage of being free of the linearity requirement, and hence, it can be applied to a wider class of nonparametric models. A detailed description of the steps involved in generating a set of bootstrap predictions can be seen in Vilar *et al.* (2010).

3. Tools for time series clustering

The dissimilarity measures presented in Section 2 and included in **TSclust** are specially useful to perform clustering on a given set of m time series $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(m)}\right\}$. Once the dissimilarity measure is selected according to the desired clustering purpose, a pairwise dissimilarity matrix $\mathcal{D}$ can be obtained and taken as starting point of a conventional clustering algorithm. Many clustering tools are available through R packages and can be used with dissimilarities generated from **TSclust**. Nevertheless, some additional tools often used in time series clustering but also useful outside this domain have been implemented in **TSclust**. In this section, we begin describing the clustering utilities implemented in **TSclust** and continue with a quick overview of some available options in R to perform, validate and visualize clustering. This overview is not intended to be a comprehensive list, but only a starting point to illustrate how to take advantage of **TSclust** by interoperating with functions from other packages.

3.1. Clustering tools in TSclust

A hierarchical clustering algorithm based on p values

The algorithm, introduced by Maharaj (2000), takes as starting point the $m \times m$ matrix $\mathcal{P} = (p_{i,j})$, whose (i,j)-th entry, $p_{i,j}$, for $i \neq j$, corresponds to the p value obtained by testing whether or not $\boldsymbol{X}_T^{(i)}$ and $\boldsymbol{X}_T^{(j)}$ come from the same generating model. Then, the algorithm proceeds in a similar way as an agglomerative hierarchical clustering based on $\mathcal{P}$, although in this case will only group together those series whose associated p values are greater than a significance level α previously specified by the user. In other words, the i-th series $\boldsymbol{X}_T^{(i)}$ will merge into a specific cluster C_k formed by the series $\left\{\boldsymbol{X}_T^{(j_1)}, \ldots, \boldsymbol{X}_T^{(j_{m_k})}\right\}$ iff $p_{i,j_l} \geq \alpha$, for all $l = 1, \ldots, m_k$. Analogously, two clusters will be joined together iff the p values of all pairs of series across the two clusters are greater than α. This algorithm behaves similar to the single linkage procedure because the dissimilarity between two clusters is the smallest dissimilarity (the greatest p value) between series in the two groups. Unlike the single linkage, two clusters will not be joined with the Maharaj's algorithm when a significant difference between a pair of series of the candidate clusters is obtained. Note also that, unlike the conventional hierarchical methods, this algorithm presents the advantage of providing automatically the number of clusters, which obviously depends on the prefixed significance level. Furthermore, the amount of compactness of each cluster can be evaluated by examining the p values within each cluster. The flow chart of this clustering algorithm can be seen in Figure 3 of Maharaj (2000), and it is available in **TSclust** by invoking the function `pvalues.clust()`.

Cluster evaluation

Two clustering evaluation criteria based on known ground-truth have been implemented in **TSclust**. One of them consists in calculating an index that measures the amount of agreement between the true cluster partition $\mathcal{G} = \{G_1, \ldots, G_k\}$ (the "ground-truth"), assumed to be known, and the experimental cluster solution $\mathcal{A} = \{A_1, \ldots, A_k\}$ obtained by a clustering method under evaluation. The similarity index $\mathit{Sim}\left(\mathcal{G}, \mathcal{A}\right)$ is defined by:

$$\mathit{Sim}\left(\mathcal{G}, \mathcal{A}\right) = \frac{1}{k} \sum_{i=1}^{k} \max_{1 \leq j \leq k} \mathit{Sim}\left(G_i, A_j\right), \tag{5}$$

where

$$Sim\left(G_i, A_j\right) = \frac{|G_i \cap A_j|}{|G_i| + |A_j|},$$

with $|\cdot|$ denoting the cardinality of the elements in the set. The index defined by Equation 5 has been used in different works on time series clustering (see Gavrilov, Anguelov, Indyk, and Motwani 2000; Kalpakis *et al.* 2001; Liao 2005; Zhang *et al.* 2006; Pértega and Vilar 2010, among others), and it is computed in **TSclust** with the function `cluster.evaluation()`.

The other evaluation method implemented in **TSclust** uses a one-nearest-neighbour (1-NN) classifier evaluated by leave-one-out cross-validation. Given the true cluster partition $\mathcal{G} = \{G_1, \ldots, G_k\}$ and a particular dissimilarity matrix $\mathcal{D}$ (i.e., a `dist` object), the 1-NN classifier assigns each series $\boldsymbol{X}_T^{(i)}$ to the element of $\mathcal{G}$ containing the nearest series, $\boldsymbol{X}_T^{(j)}$, $j \neq i$, according to the dissimilarities in $\mathcal{D}$. Then the proportion of correctly classified series is calculated. Note that this criterion directly evaluates the efficacy of the dissimilarity measure regardless of the considered clustering algorithm. This clustering evaluation procedure is intuitive, straightforward to implement, parameter-free and asymptotically optimal in the Bayes sense (Tan, Steinbach, and Kumar 2006). These reasons support its intensive use in a broad range of pattern recognition applications, including time series clustering (see, e.g., Keogh and Kasetty 2003). In **TSclust**, the evaluation based on the 1-NN classifier is available by invoking the function `loo1nn.cv()`, whose implementation allows us to deal with ties. Specifically, tied dissimilarities are solved by majority vote. If ties occur in voting, then a candidate cluster is selected at random and a warning is produced.

3.2. Some avaliable R clustering tools

As already mentioned, **TSclust** should not be seen as a stand-alone package but as an user-extensible framework whose functionality is strengthened by interfacing with existing general clustering methods. For it, a small sample of clustering utilities available in R system and helpful to interact with outcomes from **TSclust** is summarized below.

First, time series dissimilarities integrated in **TSclust** will usually be the starting point to conduct a clustering algorithm. A wide range of functions to develop different cluster algorithms are available in different R packages. For instance, primary functions to perform hierarchical clustering are `hclust()` in the package **stats** (R Core Team 2014) and `agnes()` in the package **cluster** (Maechler, Rousseeuw, Struyf, Hubert, and Hornik 2014). The **cluster** package also includes the functions `diana()` and `pam()` to carry out divisive hierarchical clustering and partitioning clustering around "medoids", respectively.

On the other hand, a variety of measures aimed at validating the results of a cluster analysis and comparing different cluster solutions are available in several R libraries. For example, the function `clValid()` in the package **clValid** (Brock, Pihur, Datta, and Datta 2008) reports a broad range of measures of clustering validation, and the function `cluster.stats()` in the package **fpc** (Hennig 2014) provides a number of distance-based statistics, which can be used for cluster validation, comparison between clustering solutions and decision about the number of clusters. Additional cluster validation techniques can be also found in the package **clv** (Nieweglowski 2013).

Some of the tools described in this section are used in the illustrative examples of Section 5.

4. Considerations on the dissimilarity measure selection

As **TSclust** provides a broad range of dissimilarity measures to perform clustering of time series, some considerations on the choice of a proper dissimilarity are given in the present section. First of all, it is worth stressing that **TSclust** should not be used as a test bed where all available dissimilarities are executed and the one reporting the "best" results is selected. In particular, what constitutes a "good" clustering is often unclear as the perception of a good clustering differs across users. Although interesting experimental comparisons of several dissimilarity measures are available in the literature (see, e.g., Keogh and Kasetty 2003; Ding, Trajcevski, Scheuermann, Wang, and Keogh 2008; Pértega and Vilar 2010), we recommend users of **TSclust** to choose the proper criterion according to the nature and specific purpose of the clustering task. Only by doing so, the cluster solution will admit an interpretation in terms of the grouping target.

A first important issue is to decide whether clustering must be governed by a "shape-based" or "structure-based" dissimilarity concept (see, e.g., Lin and Li 2009; Corduas 2010). Shape-based dissimilarity is aimed to compare the geometric profiles of the series, or alternatively, representations of them designed to reduce the dimensionality of the problem. Thus, shape-based dissimilarity principle is mainly dominated by local comparisons. On the other hand, structure-based dissimilarity is aimed to compare underlying dependence structures. Higher-level dynamic structures describing the global performance of the series must be captured and compared in this case. To shed some light on differences between both dissimilarity concepts, consider a simple and intuitive synthetic dataset of 9 time series generated from three different patterns, let us say P1, P2 and P3. All of them are displayed in Figure 2(a). The underlying patterns are identified by colour and type of line: blue solid lines for P1, red dashed lines for P2 and black dotted lines for P3. Profiles of P1 and P2 series move close within a narrow band. Therefore P1 and P2 series are the closest ones if similarity is measured in terms of proximity between geometric profiles. If, e.g., a shape-based dissimilarity like the Euclidean distance is used to perform clustering on this dataset, then the P1 and P2 series are placed together forming a mixed cluster (see Figure 2(b)). On the other hand, it can be seen that P1 and P3 have increasing/decreasing patterns closer to each other than to P2. This way P1 and P3 series become the closest ones if similarity is understood in terms of underlying correlation structures. In fact, P1 and P3 are first merged to form a cluster when clustering is carried out using a structure-based dissimilarity like d_{COR} or d_{CORT} with $k > 1$ (see Figure 2(c)).

As the interest focuses on shape-based dissimilarity, conventional distances between raw data (e.g., L_p type) or complexity-based measures (e.g., CID dissimilarity) can produce satisfactory results, although sometimes measures invariant to specific distortions of the data could be required. For instance, time series recorded in different scales will require previous normalization to cancel differences in amplitude and then match well similar shapes. In fact, conventional metrics like Minkowski, DTW or Fréchet distances may lead to common misunderstandings if the time series are not previously normalized (see illustrative examples in Rakthanmanon *et al.* 2012). Also, many biological series (e.g., recording motion capture data) are efficiently compared using dissimilarities invariant to local scaling (warping). DTW-based techniques will work specially well in these scenarios (Ding *et al.* 2008). Nevertheless, it is worth remarking that the high computational cost of dynamic time warping makes it a less desirable choice for large time series.

Overall, shaped-based dissimilarities work well with short time series but they can fail by

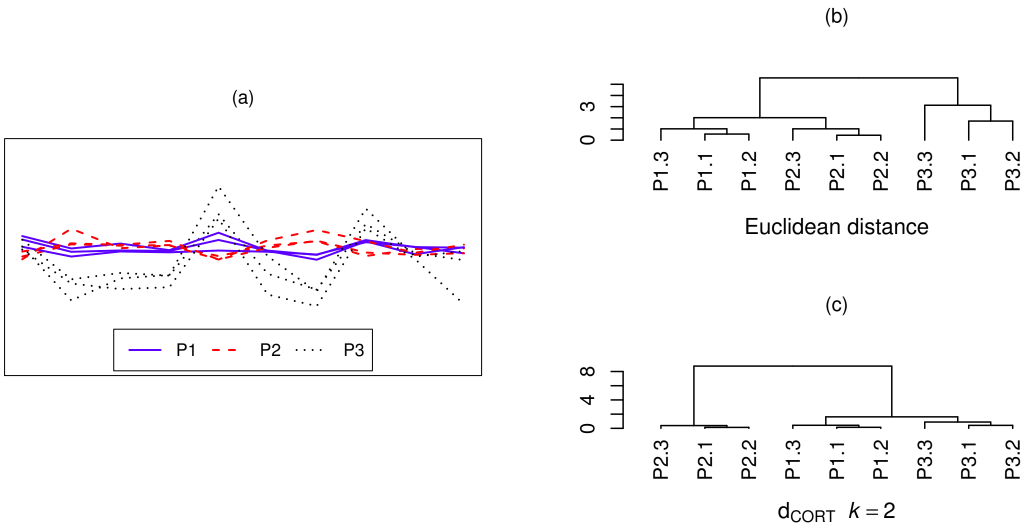

Figure 2: Realizations of 9 time series generated from different patterns P1, P2 and P3 (a). Dendrograms from clustering based on the Euclidean distance (shape-based dissimilarity, b), and d_{CORT} dissimilarity with parameter $k = 2$ (structure-based dissimilarity, c).

working with long sequences, especially when a high amount of noise or anomalous records are present. In these situations, a structure-based dissimilarity aimed to compare global underlying structures can be more appropriate. This is frequently the case when performing cluster analysis of economic or financial indicators. A range of feature- and model-based dissimilarities are included in **TSclust** to be used when a structure-based dissimilarity is required. Note that some of these dissimilarity measures assume regularity conditions for the series at hand, and users must be aware of it. For example, the model-based measures included in **TSclust** are just applicable on time series with stationary and linear underlying processes, while the prediction-based measures are free of the linearity requirement but assuming autoregressive structures of lag one.

Prediction-based dissimilarity responds to a different and easily interpretable clustering target. The real interest is not to group series showing similar shapes or underlying structures, but series with similar forecasts at a specific future time. Figure 3 illustrates this idea. Cluster analysis on the depicted time series aimed to a shape- or structure-based dissimilarity will obviously produce an entirely different result to the one coming from the study of the forecasts at a specific horizon.

Once the clustering objetives are made clear, the range of proper measures becomes more limited, and other suitable properties must be then considered. For example, most dissimilarity measures introduced in **TSclust** require setting a number of input parameters, which might produce an undesirable variability of results. Although implementation of some measures as d_{DWT} or d_{PDC} includes automatic procedures to determine these values, a careful adjustment of these parameters is recommended to find the true underlying patterns. So, practitioners should obtain background from key references to understand how adequate values can be determined in each case.

Computational complexity is also an important point. Although all measures in **TSclust** have been implemented using efficient algorithms available in R, some of them include procedures with high computational cost. For instance, measure $d_{W(LK)}$ involves numerical integration

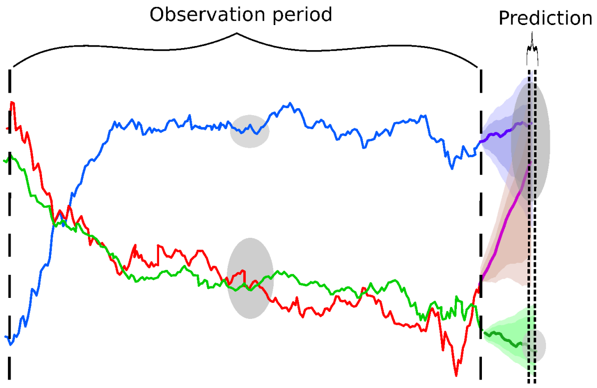

Figure 3: Illustrative example showing the two clusters obtained with any shape- or structure-based dissimilarity (evaluated on the period observation) and the two clusters derived from cluster analysis based on point predictions at a specific future time. Note that both groupings differs.

of differences between local linear smoothers computed by maximum local likelihood, which implies to solve repeatedly an optimization problem in two variables. Prediction-based measures construct kernel density estimators based on bootstrap predictions, thus increasing the computational complexity by a factor B, where B is the number of bootstrap samples. As result, both dissimilarity measures suffer of a significantly high computational complexity and could be unfeasible to perform clustering on databases including very long series. Note also that the computing times of some measures can vary largely according to the values of the dependent parameters. For instance, the autocorrelation-based distances and the Maharaj's distance have computing times $O(T \times lag.max)$ and $O(T)$, respectively, but these times become $O(T^3)$ when user specifies a matrix of weights $\mathbf{\Omega}$ in the autocorrelation-based measures or considers the extended version of the Maharaj's distance.

Attention must be also placed on the choice of the clustering algorithm, which should not distort the clustering process modifying the pre-established dissimilarity notion. In this sense, some well-known clustering methods could not work properly. For instance, the popular k-means algorithm moves each series to the cluster whose centroid is closest, recalculates the cluster centroid and repeats the assignment procedure until no time series is reassigned. Therefore k-means involves the computation of dissimilarity between time series and "centroids of a set of time series", which might not be properly defined. If for example the prediction-based metric $d_{PRED,b}$ given in Equation 4 is considered, then a centroid would be a kernel prediction density generated from an averaging of different series, and this is not reasonable at all. Similar arguments apply to other dissimilarity measures introduced in Section 2. In addition, if a hierarchical method is considered, then caution will be put when the Ward's method is chosen as linkage criterion because this algorithm is based on Euclidean distances between objects.

5. Illustrative use of TSclust package

In this section, the use of **TSclust** is illustrated by working on several sets of real and synthetic time series included in the own package. Note that the examples with real data are here considered to show the usefulness of the package **TSclust** and not to answer substantive questions in the data. Since **TSclust** provides an important number of dissimilarity measures between time series, first it is worth getting acquainted with the functions that compute these dissimilarities, which are summarized in Table 1. Given a set of time series, functions in Table 1 are mainly used to create a pairwise dissimilarity matrix (`dist` object) that is taken as base of several R clustering utilities.

5.1. An example with a synthetic dataset

For testing and comparison purposes, **TSclust** includes a collection of eighteen synthetic time series available by loading `synthetic.tseries`. More precisely, `synthetic.tseries` consists of three partial realizations of length $T = 200$ of each of the six first order autoregressive models enumerated in Table 2.

In all cases, ε_t are i.i.d. Gaussian random variables with zero mean and unit variance. These models have been selected to include both linear and non-linear structures. Model M1 is an AR(1) process with moderate autocorrelation. Model M2 is a bilinear process with approximately quadratic conditional mean, and thus, strongly non-linear. Model M3 is an exponential autoregressive model with a more complex non-linear structure although very close to linearity. Model M4 is a self-exciting threshold autoregressive model with a relatively strong non-linearity. Finally, Models M5, a general non-linear autoregressive model, and M6, a smooth transition autoregressive model, present a weak non-linear structure. Therefore, the selected models represent different nonlinear structures for the conditional mean, thus providing a valuable scenario to examine the performance in clustering of the dissimilarity measures implemented in **TSclust**. In fact, some of these models were previously considered by Vilar *et al.* (2010) in a broad simulation study conducted to attain this objective. Here, our interest is mainly illustrative and for this reason our analysis is limited to this set of series. Indeed, a more extensive analysis including a large number of simulated replicates should be required to state rigorous conclusions.

We proceed as follows. Series in `synthetic.tseries` are subjected to clustering by using different dissimilarity measures and several clustering algorithms. Assuming that the clustering is governed by the similarity between underlying models, the "true" cluster solution is given by the six clusters involving the three series from the same generating model. Then, the experimental cluster solutions are compared with the true cluster solution by using the cluster similarity function `cluster.evaluation()` included in **TSclust**.

First steps are loading both the package and dataset and creating the true solution.

```
R> library("TSclust")
R> data("synthetic.tseries")
R> true_cluster <- rep(1:6, each = 3)
```

We start testing the dissimilarity measure d_{IP}, that computes the L_2 distance between integrated periodograms and can be evaluated in **TSclust** by invoking `diss.INT.PER()`. The wrapper function `diss()` is provided as an easy way to apply any of the **TSclust** dissimilarity functions to a matrix of time series data. Function `diss()` takes a dataset and a string specifying the **TSclust** dissimilarity to be used, and returns a `dist` object with all the pairwise dissimilarities. A dataset can be introduced as a `mts` object, or alternatively as a `matrix`,

Dissimilarity measures	Function in **TSclust**
Model free approaches	
Based on raw data	
d_{L_2}(Euclidean)	`diss.EUCL`
d_F	`diss.FRECHET`
d_{DTW}	`diss.DTW`
Covering both proximity on values and on behavior	
d_{CORT}	`diss.CORT`
Based on correlations	
$d_{COR.1}$	`diss.COR`, `beta = NULL`
$d_{COR.2}$	`diss.COR` and a specified value for `beta`
Based on simple and partial autocorrelations	
d_{ACF}	`diss.ACF`
d_{ACFU}	`diss.ACF` without parameters
d_{ACFG}	`diss.ACF`, `p` $\neq 0$
d_{PACF}	`diss.PACF`
d_{PACFU}	`diss.PACF` without parameters
d_{PACFG}	`diss.PACF`, `p` $\neq 0$
Based on periodograms	
d_P	`diss.PER`
d_{NP}	`diss.PER`, `normalize = TRUE`
d_{LNP}	`diss.PER`, `normalize = TRUE`, `logarithm = TRUE`
d_{IP}	`diss.INT.PER`
Based on nonparametric spectral estimators	
$d_{W(LS)}$	`diss.SPEC.LLR`, `method = "LS"`
$d_{W(LK)}$	`diss.SPEC.LLR`, `method = "LK"`
d_{GLK}	`diss.SPEC.GLK`
d_{ISD}	`diss.SPEC.IDS`
Based on the discrete wavelet transform	
d_{DWT}	`diss.DWT`
Based on symbolic representation	
$d_{MINDIST.SAX}$	`diss.MINDIST.SAX`
Model-based approaches	
d_{PIC}	`diss.AR.PIC`
d_{MAH}	`diss.AR.MAH`
d_{MAHext}	`diss.AR.MAH`, `dependence = TRUE`
$d_{LCP.Cep}$	`diss.AR.LPC.CEPS`
Complexity-based approaches	
d_{CID}	`diss.CID`
d_{PDC}	`diss.PDC`
d_{CDM}	`diss.CDM`
d_{NCD}	`diss.NCD`
Prediction-based approach	
$d_{PRED,h}$	`diss.PRED`

Table 1: Dissimilarity measures implemented in **TSclust** (see Section 2 for details).

M1	AR	$X_t = 0.6X_{t-1} + \varepsilon_t$
M2	Bilinear	$X_t = (0.3 - 0.2\varepsilon_{t-1}) X_{t-1} + 1.0 + \varepsilon_t$
M3	EXPAR	$X_t = \left(0.9 \exp\left(-X_{t-1}^2\right) - 0.6\right) X_{t-1} + 1.0 + \varepsilon_t$
M4	SETAR	$X_t = (0.3X_{t-1} + 1.0)\, I\,(X_{t-1} \geq 0.2) - (0.3X_{t-1} - 1.0)\, I\,(X_{t-1} < 0.2) + \varepsilon_t$
M5	NLAR	$X_t = 0.7\,\lvert X_{t-1}\rvert\,(2 + \lvert X_{t-1}\rvert)^{-1} + \varepsilon_t$
M6	STAR	$X_t = 0.8X_{t-1} - 0.8X_{t-1}\,(1 + \exp(-10X_{t-1}))^{-1} + \varepsilon_t$

Table 2: Generating models for time series in `synthetic.tseries`.

`list` or `data.frame` object. Additional arguments required by the particular dissimilarity must be also passed to `diss()` function. In our example, `diss()` is used as follows.

```
R> IP.dis <- diss(synthetic.tseries, "INT.PER")
```

Now, the object `IP.dis` contains the dissimilarity matrix computed by applying d_{IP} to each pair of series in `synthetic.tseries` and can be taken as starting point for several conventional clustering methods. For example, we use `hclust()` of the base package **stats** to conduct a hierarchical clustering algorithm. We get the six cluster solution from the hierarchical clustering using the `cutree()` function of **stats**.

```
R> IP.hclus <- cutree(hclust(IP.dis), k = 6)
R> cluster.evaluation(true_cluster, IP.hclus)

[1] 0.784127
```

As the correct number of clusters is known, a partitive clustering technique as the popular Partitioning Around Medoids (PAM) algorithm is also considered. We use the function `pam()` in package **cluster**.

```
R> library("cluster")
R> IP.pamclus <- pam(IP.dis, k = 6)$clustering
R> cluster.evaluation(true_cluster, IP.pamclus)

[1] 0.8888889
```

Recall that the output of `cluster.evaluation()` lies between 0 and 1 and admits a simple interpretation: the closer it is to 1, the higher is the agreement between the true and experimental partitions. Therefore, it can be concluded that the model-free measure d_{IP} produces good results in this case, showing the best performance as the PAM algorithm is used. In fact, the cluster solution obtained with the PAM algorithm only classifies incorrectly two series.

```
R> IP.pamclus

   AR.1     AR.2     AR.3 BILIN.1 BILIN.2 BILIN.3 EXPAR.1 EXPAR.2 EXPAR.3
      1        1        1       2       2       3       4       4       4
SETAR.1 SETAR.2 SETAR.3  NLAR.1  NLAR.2  NLAR.3  STAR.1  STAR.2  STAR.3
      5        5        5       3       3       2       6       6       6
```

Any dissimilarity function can be tested in a similar way. Consider for example the dissimilarity based on the simple autocorrelations d_{ACF}.

```
R> ACF.dis <- diss(synthetic.tseries, "ACF", p = 0.05)
R> ACF.hclus <- cutree(hclust(ACF.dis), k = 6)
R> cluster.evaluation(true_cluster, ACF.hclus)

[1] 0.7444444

R> ACF.pamclus <- pam(ACF.dis, k = 6)$clustering
R> cluster.evaluation(true_cluster, ACF.pamclus)

[1] 0.6666667

R> ACF.hclus

   AR.1    AR.2    AR.3 BILIN.1 BILIN.2 BILIN.3 EXPAR.1 EXPAR.2 EXPAR.3
      1       1       1       2       2       2       3       3       4
SETAR.1 SETAR.2 SETAR.3  NLAR.1  NLAR.2  NLAR.3  STAR.1  STAR.2  STAR.3
      5       5       2       2       2       2       5       6       6
```

The ACF-based dissimilarity leads to worse results, with similarity indexes below 0.75. Here the hierarchical algorithm outperforms the PAM algorithm, but only the group formed by the AR(1) series is correctly identified.

Let us now consider the model-based dissimilarity $d_{MAH,p}$ introduced by Maharaj (2000) which is implemented in the function `diss.AR.MAH()`. As $d_{MAH,p}$ produces p values obtained by testing the equality of generating ARMA-models, we can use the hierarchical algorithm `pvalues.clust()` based on p values described in Section 3.

```
R> AR.MAH.PVAL.dis <- diss(synthetic.tseries, "AR.MAH")$p_value
R> AR.MAH.PVAL.clus <- pvalues.clust(AR.MAH.PVAL.dis, significance = 0.05)
R> cluster.evaluation(true_cluster, AR.MAH.PVAL.clus)

[1] 0.6901515

R> AR.MAH.PVAL.clus

[1] 2 2 2 1 1 1 3 3 1 4 4 1 1 1 1 4 4 4
```

The number of clusters is automatically established by the clustering algorithm depending on the threshold significance specified in `significance`. Using a 0.05 significance level only four clusters are identified. The AR(1) series are correctly placed forming one group, the exponential autoregressive series (EXPAR) form other group although one of them is erroneously not included, and the two remaining groups mainly place together the SETAR and STAR structures and the bilinear and NLAR series, respectively. Indeed, this poor result is expected because $d_{MAH,p}$ is aimed to assume that the generating processes are ARMA models. Note that by increasing the threshold significance, the pairwise equality tests become less conservative, and thus a greater number of clusters can be found. For instance, setting `significance = 0.6` in our example, six clusters are found.

```
R> pvalues.clust(AR.MAH.PVAL.dis, significance = 0.6)

[1] 4 4 4 1 1 1 5 5 2 3 3 3 1 1 1 6 6 6
```

Finally, we consider a d_W-type dissimilarity measure based on nonparametric spectral approximations such as defined in Equation 1.

```
R> LLR.dis <- diss(synthetic.tseries, "SPEC.LLR", method = "LK", n = 500)
R> LLR.pam <- pam(LLR.dis, k = 6)$clustering
R> cluster.evaluation(true_cluster, LLR.pam)

[1] 0.8333333

R> LLR.pam

   AR.1    AR.2    AR.3 BILIN.1 BILIN.2 BILIN.3 EXPAR.1 EXPAR.2 EXPAR.3
      1       1       1       2       2       3       4       4       4
SETAR.1 SETAR.2 SETAR.3  NLAR.1  NLAR.2  NLAR.3  STAR.1  STAR.2  STAR.3
      5       5       5       5       2       2       6       6       6
```

As expected, a reasonably good result (only three series are misclassified) is obtained because of $d_{W(LK)}$ is free of the linearity assumption, and hence it is especially robust to perform clustering in the present framework.

5.2. Clustering interest rate series

Our second example deals with long-term interest rate data included in **TSclust** and available by loading `interest.rates`. This dataset is formed by 18 time series of length $T = 215$ representing monthly long-term interest rates (10-year bonds), issued in percentages per annum, for seventeen countries and the Economic and Monetary Union (EMU). The observation period goes from January 1995 to November 2012 (OECD source: `http://www.oecd.org/`). Long-term government bonds are the instrument whose yield is used as the representative "interest rate" for each area, and hence these indicators are a vital tool of monetary policy and are taken into account when dealing with variables like investment, inflation, and unemployment. In particular, one of the convergence criteria required for all countries in the euro area is related to this indicator. A lattice plot of the series in `interest.rates` is shown in Figure 4.

Data in `interest.rates` are here used to perform clustering from two different points of view. First, the original series are clustered in accordance with the performance of the predictions at a given horizon. Identifying groups of countries with similar predictions for their long-term interest rates is a meaningful objective due to the influence of these indicators on the monetary policy actions. In any case, our application just intends illustrating the usefulness of **TSclust** to achieve this objective without studying this interesting issue in depth. The proper function to carry out prediction-based clustering is `diss.PRED()`, which computes the L_1 distance between bootstrap approximations of the h-steps-ahead forecast densities for two given time series (see $d_{PRED,h}$ in Equation 4). Suppose we wish to group together the

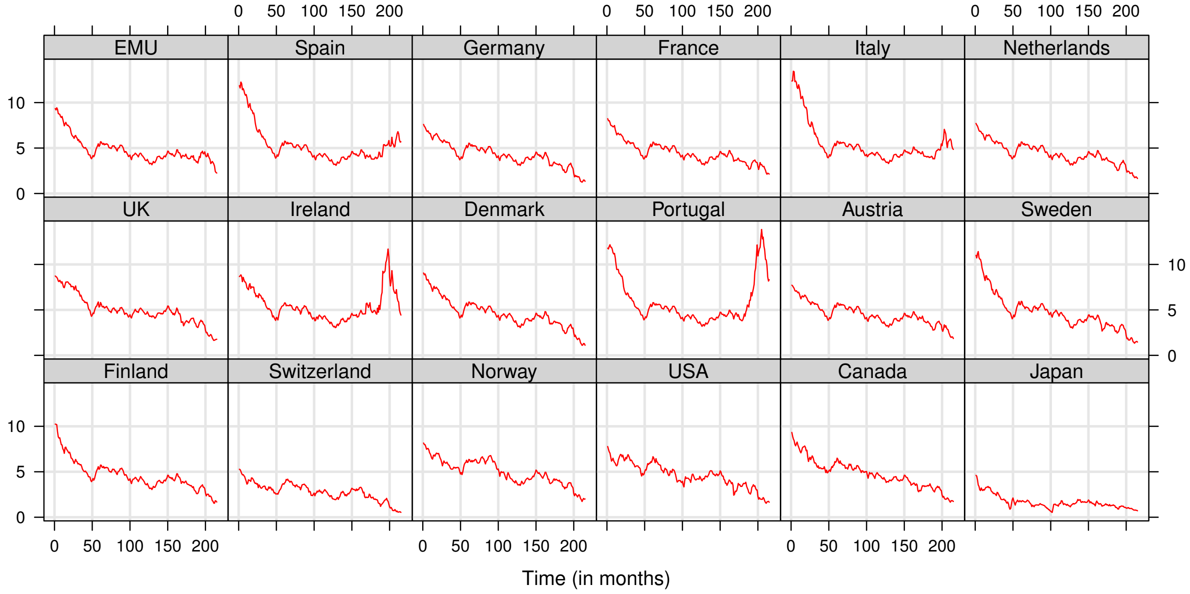

Figure 4: Monthly long-term interest rate series (10-year bonds), issued in percentages per annum, over the period January 1995 to November 2012.

countries with similar six-months-ahead predictions ($h = 6$). As the bootstrap mechanism requires stationarity and the time series under study are clearly non-stationary, the original series are previously transformed by using logarithms (if required) and taking an appropriate number of regular differences. Bootstrap prediction-paths are generated from the transformed series, and then the prediction-paths are back-transformed to obtain bootstrap predictions for the original series. All these steps are automatically carried out by `diss.PRED()`. The required arguments for this function are the horizon of interest, `h`, the number of bootstrap resamples, `B`, and how transformation must be carried out, that is if logarithms are or not taken (`logarithms.x` and `logarithms.y`) and the number of regular differences to be applied (`differences.x` and `differences.y`). We start comparing the predictions for Finland and USA (columns 13 and 16 of `interest.rates`, respectively).

```
R> data("interest.rates")
R> diss.PRED(interest.rates[, 13], interest.rates[, 16], h = 6, B = 2000,
+    logarithm.x = TRUE, logarithm.y = TRUE, differences.x = 1,
+    differences.y = 1, plot = TRUE)$L1dist

[1] 0.2248442
```

The plot generated by `diss.PRED()` is shown on the right part of Figure 6. Note that the prediction densities have similar means, but differ in their shapes (the forecast density of Finland presents a higher kurtosis), and this feature is captured by using the L_1 distance between prediction densities (taking the value 0.2248442 in this particular case).

When `diss.PRED()` is used from the wrapper `diss()`, arguments indicating the transformation features are passed from two new arguments, `logarithms` and `differences`, where the proper transformation values for all series are gathered together preserving the order in the

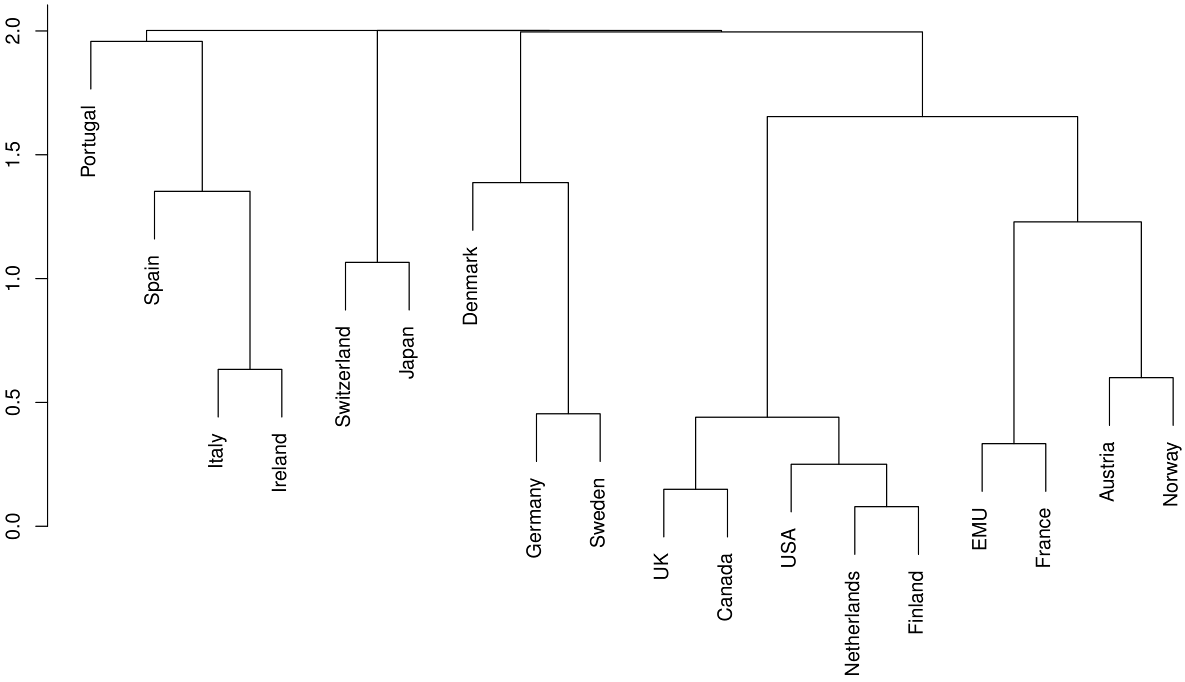

Figure 5: Dendrogram for `interest.rate` dataset based on the six-months-ahead prediction density estimates by using the complete linkage.

original dataset. For instance, assume that the recommended transformation for all series in `interest.rates` is to take one difference of the logarithm, then the dissimilarity matrix based on the six-months-ahead prediction density estimates can be obtained as follows.

```
R> diffs <- rep(1, ncol(interest.rates))
R> logs  <- rep(TRUE, ncol(interest.rates))
R> dpred <- diss(interest.rates, "PRED", h = 6, B = 1200, logarithms = logs,
+    differences = diffs, plot = TRUE)
```

Matrix `dpred$dist` is now used to perform clustering.

```
R> hc.dpred <- hclust(dpred$dist)
R> plot(hc.pred)
```

Figure 5 shows the resulting dendrogram. The four cluster solution seems to determine groups reasonably well-separated, namely $\mathcal{C}_1$ = {Ireland, Italy, Spain, Portugal}, $\mathcal{C}_2$ = {Germany, Sweden, Denmark}, $\mathcal{C}_3$ = {Switzerland, Japan}, and the fourth cluster $\mathcal{C}_4$ formed by the remaining countries. Indeed, if a finer partition identifying more compact groups is required, then $\mathcal{C}_4$ can be split into two homogeneous subclusters, and Portugal (and in a lesser extension Spain and Denmark) would leave their groups to appear like isolated points. To gain insight into the clustering, all the estimated forecast densities have been jointly depicted in Figure 6 by using the option `plot = TRUE` in the function call.

Figure 6 allows us to characterize the clusters in terms of the prediction densities for their memberships. Cluster $\mathcal{C}_1$ brings together the four countries more affected by the Eurozone

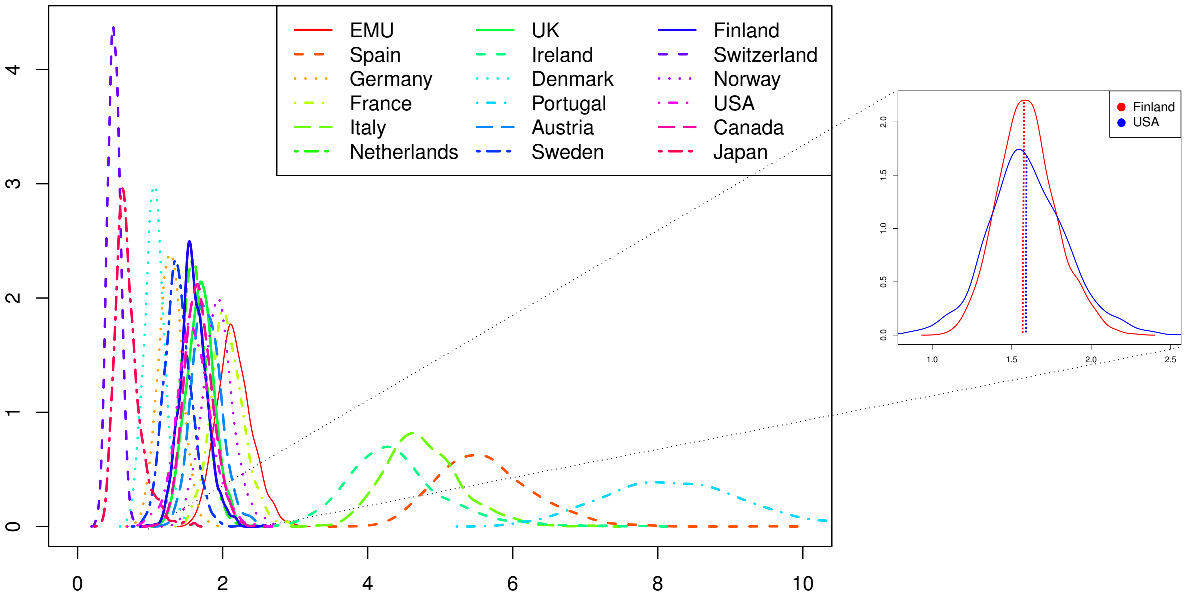

Figure 6: Six-months-ahead prediction density estimates for long-term interest rates. Densities for Finland and USA are zoomed at the right part.

crisis, presenting the largest predictions for their long-term interest rates. All of them show a high amount of variability in their prediction densities (especially Portugal), thus increasing the uncertainty for their future interest rates. By contrast, Japan and Switzerland, forming $\mathcal{C}_3$, are the countries with the lowest predictions and less uncertainty. $\mathcal{C}_1$ and $\mathcal{C}_3$ are perfectly separated and constitute very stable clusters because no overlaps are given between densities from these two groups. Clusters $\mathcal{C}_2$ and $\mathcal{C}_4$ collect countries with intermediate predictions, also well-separated from $\mathcal{C}_1$ and $\mathcal{C}_3$. Denmark, Germany and Sweden, forming $\mathcal{C}_2$, present predictions close to the lowest ones from $\mathcal{C}_3$, especially Denmark, and the countries in $\mathcal{C}_4$ show somewhat higher predictions, but in any case significantly lower than the countries in $\mathcal{C}_1$. It is worthy to remark that a classification based on pointwise predictions differs from the classification aimed to prediction densities. For example, Portugal and Spain are clearly isolated points if a pointwise prediction approach is considered, while they are in the same cluster with the density criterion.

Now, a different clustering problem based on the same interest rate dataset is posed. Specifically, we are interested in performing cluster analysis with the series transformed by taking the first difference of the logarithms, i.e., by using series $\boldsymbol{Y}_T^{(i)}$, $i = 1, \ldots, 24$, given by $\log X_t^{(i)} - \log X_{t-1}^{(i)}$, $t = 2, \ldots, T$. Actually $\boldsymbol{Y}_T^{(i)}$ approximately measures the percentage changes in the interest rates from month to month, and therefore this new cluster analysis intends to group together those countries with similar monthly increases in their interest rates. A lattice plot of the new time series is shown in Figure 7.

The clustering aim is now to identify similar dependence structures hidden behind the wiggly and noisy profiles observed in Figure 7. Note that nonstationarity of the original series has been removed by taking differences, and therefore dissimilarity measures constructed under the stationarity assumption can be used. For illustrative purposes, we perform hierarchical clustering with complete linkage and use several dissimilarity measures implemented in

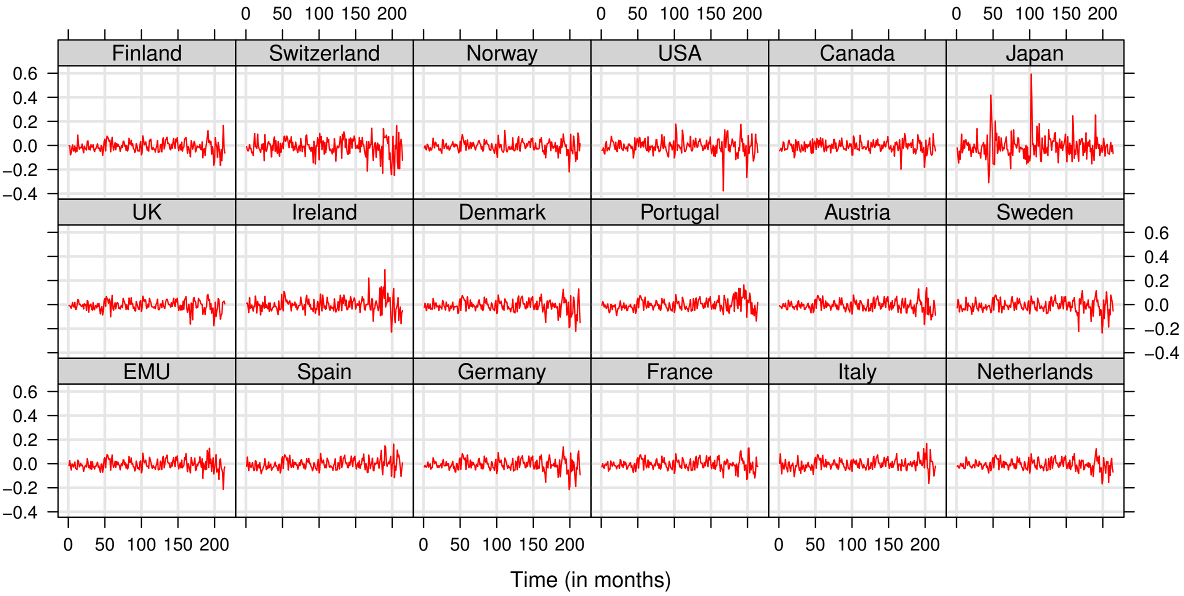

Figure 7: Long-term interest rates transformed by taking the first difference of the logarithms.

TSclust, namely the dissimilarities based on the simple autocorrelation functions, the logarithm of the normalized periodograms and the AR-metric introduced by Piccolo. The average Silhouette coefficients were examined for several k-cluster solutions and it was observed that a number of $k = 5$ groups yields a reasonably compact solution. First, the time series are transformed and a matrix for storing the 5-cluster solutions is created.

```
R> relative.rate.change <- diff(log(interest.rates), 1)
R> Five.cluster.sol <- matrix(0, nrow = 18, ncol = 3)
R> colnames(Five.cluster.sol) <- c("ACF", "LNP", "PIC")
R> rownames(Five.cluster.sol) <- colnames(relative.rate.change)
```

The following code leads to the 5-cluster solutions for the three dissimilarity measures.

```
R> Five.cluster.sol[, 1] <- cutree(hclust(diss(relative.rate.change,
+    "ACF", p = 0.05)), k = 5)
R> Five.cluster.sol[, 2] <- cutree(hclust(diss(relative.rate.change, "PER",
+    normalize = TRUE, logarithm = TRUE)), k = 5)
R> Five.cluster.sol[, 3] <- cutree(hclust(diss(relative.rate.change,
+    "AR.PIC")), k = 5)
R> Five.cluster.sol
```

```
            ACF LNP PIC
EMU           1   1   1
Spain         1   1   1
Germany       2   2   1
France        2   1   2
Italy         1   1   1
```

```
Netherlands   2   2   1
UK            2   2   3
Ireland       1   3   1
Denmark       2   2   2
Portugal      3   3   4
Austria       2   1   2
Sweden        2   2   1
Finland       2   2   2
Switzerland   4   4   4
Norway        2   2   1
USA           2   2   3
Canada        2   2   3
Japan         5   5   5
```

Similar results are obtained with the three dissimilarities. Japan, Switzerland and Portugal are isolated objects. Japan clearly shows the greatest variability in the relative changes of their interest rates. Switzerland also presents an important variability and a large sequence of months with substantial negative growth. Portugal is closer to the rest of countries although presents very few months of large decay. With respect to the remaining countries, it is complex to find out substantial differences, although it is worth pointing out that four sets of countries are located together with the three considered dissimilarity criteria, namely {Spain, Italy, EMU}, {Germany, Netherlands, Sweden, Norway}, {Finland, Denmark} and {USA, Canada, UK}.

5.3. Matching pairs of times series from different domains

Now, we consider a dataset formed by a collection of 18 pairs of time series covering different domains, including finance, science, medicine, industry, etc. This dataset was presented by Keogh *et al.* (2004) to evaluate different clustering procedures and has been later considered by other authors (Lin and Li 2009; Brandmaier 2011, among others). The dataset is freely available at `http://www.cs.ucr.edu/~eamonn/SIGKDD2004/`, and included in **TSclust** as `paired.tseries`. Each pair of series was selected from a different domain in datasets from the University of California Riverside (UCR) Time Series Archive (Keogh *et al.* 2011). For example, the dataset includes the series *thoracic* and *abdominal* from the Fetal-ECG archive, annual power demand series for two different months *Power: April-June* and *Power: Jan-March*, from the Italy-Power-Demand archive, and so on. The 36 time series are displayed in Figure 8, where a different colour is used to identify each pair. Note that some of the distinct pairs present clearly different profiles.

One of the experiments described by Keogh *et al.* (2004) consisted in performing agglomerative hierarchical cluster analysis starting from different pairwise dissimilarities. The main objective was to identify the 18 pairs at the lowest level of the tree, assuming that a clustering in which each pair of time series is closest to each other is the ground truth. For this purpose, the quality of clustering is evaluated with a measure Q defined by the number of correct pairings at the lowest clustering level divided by 18. A perfect result is attained when $Q = 1$, while the authors argue that $Q = 0$ would be expected for a random clustering due to the high number of possible dendrograms with 36 objects (greater than 3×10^{49}).

Package **TSclust** is a useful tool to carry out this type of experiments in a simple way. For

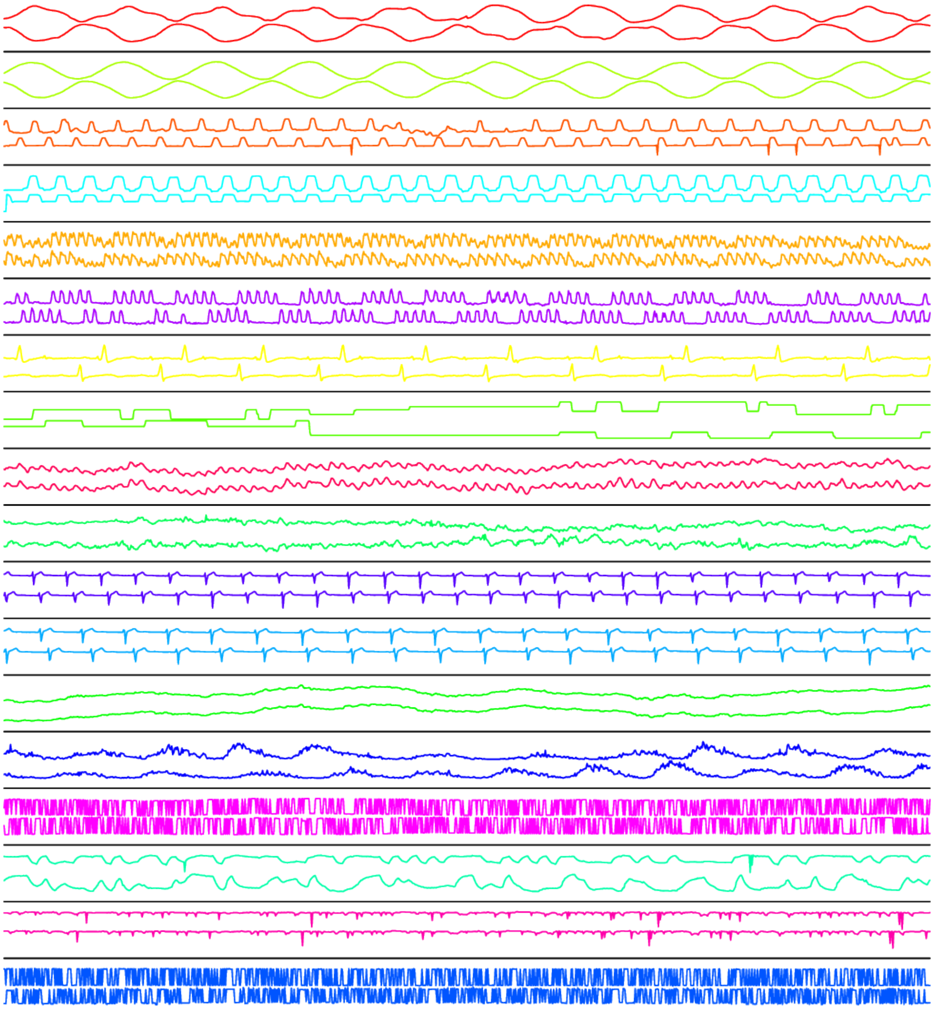

Figure 8: Eighteen pairs of time series from the UCR Time Series Archive (Keogh *et al.* 2011) subjected to hierarchical clustering in Keogh *et al.* (2004).

illustrative purposes, some of the dissimilarity measures are evaluated using functions of **TSclust**. We begin by loading the dataset and creating the "ground-truth" solution. Since `hclust()` will be used to perform hierarchical clustering, the i-th row of the output `merge` describes the merging of clusters at step i of the hierarchical process. A negative element j in the row means that observation $-j$ was merged at this stage. Hence, a row with negative entries indicates agglomeration of singletons, and regarding the order of the series in the database, this row represents one of the desired pairings when takes the form $(-j, -(j+1))$, with j odd. For this reason, `true_pairs` is defined as the 2×36 matrix whose j-th column is the pair $(-j, -(j+1))$. The Q-measure introduced by Keogh *et al.* (2011) is then evaluated by counting the number of matches between `true_pairs` and `hclust$merge`.

```
R> data("paired.tseries")
R> true_pairs <- data.frame(-matrix(1:36, nrow = 2, byrow = FALSE))
```

A glance at Figure 8 suggests substantial differences between shapes and complexity levels of some pairs of series. Thus, a conventional shape-based dissimilarity or some complexity-based dissimilarity could produce good results. We begin examining up to three dissimilarities, namely the Euclidean distance, evaluating point-to-point discrepancy and so emphasizing local shape differences, d_{CID}, addressed to compare shape complexities, and d_{PDC}, measuring divergence between permutation distributions of the series.

```
R> deucl <- diss(paired.tseries, "EUCL")
R> hceucl <- hclust(deucl, "complete")
R> Qeucl <- sum(true_pairs %in% data.frame(t(hceucl$merge)))/18
R> Qeucl

[1] 0.2777778

R> dcid <- diss(paired.tseries, "CID")
R> hccid <- hclust(dcid, "complete")
R> Qcid <- sum(true_pairs %in% data.frame(t(hccid$merge)))/18
R> Qcid

[1] 0.6111111

R> dpdc <- diss(paired.tseries, "PDC")
R> hcpdc <- hclust(dpdc, "complete")
R> Qpdc <- sum(true_pairs %in% data.frame(t(hcpdc$merge)))/18
R> Qpdc

[1] 0.7222222
```

Five pairs of series ($Q = 0.27$) are successfully clustered together at the lowest level of the tree by using the Euclidean distance. Notice that this performance is significantly better than a random clustering ($Q = 0$), thereby showing the difficulty to separate correctly all the pairs. In fact, Keogh *et al.* (2004) argue that more than 3/4 of the tested clustering approaches yielded the worst possible score $Q = 0$. It is relevant that the parameter-free CID dissimilarity substantially improves the result, identifying correctly 11 pairs. The best result, 13 out of 18 pairs, corresponds to the PDC approach using the embedding dimension automatically chosen by the heuristic procedure proposed by Brandmaier (2011), $m = 5$ in this case.

Next, the compression-based dissimilarity `diss.CDM` is examined using `gzip` as compression algorithm. Alternative options in **TSclust** to determine the compression algorithm are `bzip2`, `xzip`, and `min`. If `min` is selected, the other three options will be tested, selecting the one that gives the best compression.

```
R> dcdm <- diss(paired.tseries, "CDM", "gz")
R> hccdm <- hclust(dcdm, "complete")
R> Qcdm <- sum(true_pairs %in% data.frame(t(hccdm$merge)))/18
R> Qcdm

[1] 0.5
```

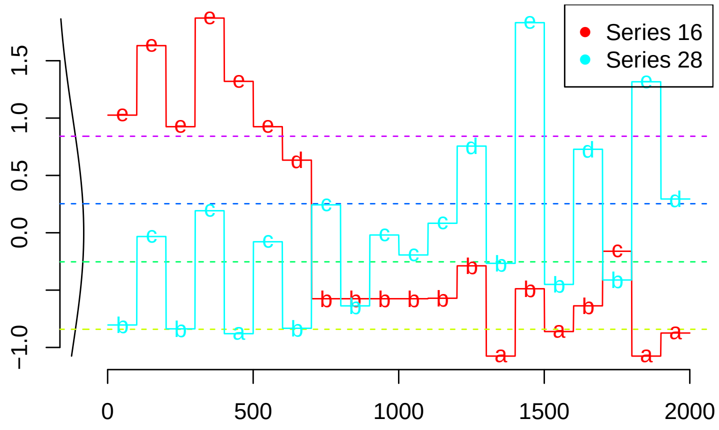

Figure 9: Plot produced by `plot.SAX` showing both PAA reduction and SAX symbolization for two series in `paired.tseries`. Dotted lines indicate the breakpoints producing equal-sized areas under a standard Gaussian curve. Based on these breakpoints, PAA coefficients (horizontal segments) are mapped into symbols (letters from "a" to "e") with equiprobability. Here, the reduced dimension $w = 20$ and the alphabet size $\alpha = 4$ produce the SAX representations `eeeeeedbbbbbbababcaa` and `bcbcacbcbcccdbebdbed` for series 16 and 28, respectively.

Dissimilarity `diss.CDM` with `gzip` compression gets 9 out of 18 pairs correctly. Nevertheless, the compression-based procedures can be affected by numerical issues (Keogh *et al.* 2004) and a symbolic representation of the series can help overcome these problems. Although there are many different symbolic approximations of time series in the literature (see, e.g., the review of symbolic analysis of experimental data provided by Daw *et al.* 2003), the Symbolic Aggregate ApproXimation (SAX) representation has been implemented in **TSclust** because allows both dimensionality reduction and lower bounding (Lin *et al.* 2003). SAX consists of a z-normalization, a PAA reduction, and a further transformation where PAA representations are mapped to symbols with equiprobability, thus obtaining the SAX representation (see Section 2.1). Both PAA representation and SAX symbolization are performed in **TSclust** using the routines `PAA` and `convert.to.SAX.symbol`, respectively. A graphical illustration of the combined result of these procedures is provided by `SAX.plot` function. See Figure 9 generated by the code below.

```
R> testseries <- as.ts(cbind(paired.tseries[, 16], paired.tseries[, 28]))
R> colnames(testseries) <- c("Series 16", "Series 28")
R> SAX.plot(testseries, w = 20, alpha = 5)
```

To illustrate how the compression-based dissimilarity `diss.CDM` can substantially improve by using the symbolic SAX representation, a prior search for the input parameters was conducted. As result, a perfect clustering $Q = 1$ (as in Keogh *et al.* 2004) is attained if the following code lines are executed.

```
R> tsersax <- apply(paired.tseries, 2, function(x) {
+    x <- (x - mean(x)) / sd(x)
+    x <- PAA(x, w = 343)
+    convert.to.SAX.symbol(x, alpha = 27)})
```

```
R> tsersax <- as.ts(tsersax)
R> dcdmsymb <- diss(tsersax, "CDM", "gz")
R> hccdmsymb <- hclust(dcdmsymb, "ward")
R> Qcdmsymb <- sum(true_pairs %in% data.frame(t(hccdmsymb$merge)))/18
R> Qcdmsymb

[1] 1
```

Note that other dissimilarities may attain a Q-optimal classification if a search of hyperparameters is performed. In particular, $Q = 1$ is also retrieved using PDC as follows.

```
R> dpdc_m5t8 <- diss(paired.tseries, "PDC", m = 5, t = 8)
R> hcpdc_m5t8 <- hclust(dpdc_m5t8, "complete")
R> Qpdc_m5t8 <- sum(true_pairs %in% data.frame(t(hcpdc_m5t8$merge)))/18
R> Qpdc_m5t8

[1] 1
```

6. Conclusions

A key issue in time series clustering is the choice of a suitable measure to assess the dissimilarity between two time series data. A large number of well-established and peer-reviewed dissimilarity measures between time series have been proposed in the literature, and a wide range of them are included in the R package **TSclust** presented in this work. **TSclust** is mainly an attempt to integrate different time series dissimilarity criteria in a single software package to check and compare their behavior in clustering. The main motivation behind this package is that, to our knowledge, no previous packages are available targeting the problem of clustering time series, except for the **pdc** package (Brandmaier 2014), which is mainly focused on the permutation distribution clustering. Nevertheless, demand for a package of these features is supported by the increasing number of references and applications in different fields. A brief description and key references of the dissimilarity measures implemented in **TSclust** are included in the first part of this paper to give insight into their rationale. By using some real and synthetic dataset examples, the capabilities and usefulness of the package are illustrated. Several scenarios with different clustering objectives have been proposed to support the need of considering different dissimilarity measures. In this sense, some general considerations on how to choose the proper dissimilarity are also discussed. The package **TSclust** is under continuous development with the purpose of incorporating new dissimilarity criteria and clustering utilities in time series framework.

Acknowledgments

The authors wish to thank the three anonymous reviewers and the Associate Editor for their helpful comments and valuable suggestions, which have allowed us to improve the quality of this work. The work of J.A. Vilar was supported by the Spanish grant MTM2011-22392 from the Ministerio de Ciencia y Tecnología.

References

Alonso AM, Berrendero JR, Hernandez A, Justel A (2006). "Time Series Clustering Based on Forecast Densities." *Computational Statistics & Data Analysis*, **51**, 762–776.

Amari S (2007). "Integration of Stochastic Models by Minimizing α-Divergence." *Neural Computation*, **19**(10), 2780–2796.

Bagnall AJ, Janacek G, de la Iglesia B, Zhang M (2003). "Clustering Time Series from Mixture Polynomial Models with Discretised Data." In *Proceedings of the 2nd Australasian Data Mining Workshop*, pp. 105–120. University of Technology Sydney.

Batista GEAPA, Wang X, Keogh EJ (2011). "A Complexity-Invariant Distance Measure for Time Series." In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, SDM'11, pp. 699–710. SIAM, Mesa.

Berndt DJ, Clifford J (1994). "Using Dynamic Time Warping to Find Patterns in Time Series." In *KDD Workshop*, pp. 359–370.

Bohte Z, Cepar D, Košmelj K (1980). "Clustering of Time Series." In MM Barritt, D Wishart (eds.), *COMPTSTAT 80, Proceedings in Computational Statistics*, pp. 587–593. Physica-Verlag, Heidelberg.

Brandmaier AM (2011). *Permutation Distribution Clustering and Structural Equation Model Trees*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany.

Brandmaier AM (2014). ***pdc**: Permutation Distribution Clustering*. R package version 1.0.1, URL http://CRAN.R-project.org/package=pdc.

Brock G, Pihur V, Datta S, Datta S (2008). "**clValid**: An R Package for Cluster Validation." *Journal of Statistical Software*, **25**(4), 1–22. URL http://www.jstatsoft.org/v25/i04/.

Caiado J, Crato N, Peña D (2006). "A Periodogram-Based Metric for Time Series Classification." *Computational Statistics & Data Analysis*, **50**(10), 2668–2684.

Casado de Lucas D (2010). *Classification Techniques for Time Series and Functional Data*. Ph.D. thesis, Universidad Carlos III de Madrid.

Chan FK, Fu AW (1999). "Efficient Time Series Matching by Wavelets." In *Proceedings of 15th International Conference on Data Engineering (ICDE '99)*, pp. 126–133.

Chan FK, Fu AW, Yu C (2003). "Haar Wavelets for Efficient Similarity Search of Time-Series: With and without Time Warping." *IEEE Transactions on Knowledge and Data Engineering*, **15**(3), 686–705.

Cilibrasi R, Vitányi PMB (2005). "Clustering by Compression." *IEEE Transactions on Information Theory*, **51**, 1523–1545.

Constantine W, Percival D (2014). ***wmtsa**: Wavelet Methods for Time Series Analysis*. R package version 2.0-0, URL http://CRAN.R-project.org/package=wmtsa.

Corduas M (2010). “Mining Time Series Data: A Selective Survey.” In F Palumbo, CN Lauro, MJ Greenacre (eds.), *Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 355–362. Springer-Verlag.

Daw CS, Finney CEA, Tracy ER (2003). “A Review of Symbolic Analysis of Experimental Data.” *Review of Scientific Instruments*, **74**(2), 915–930.

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008). “Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures.” *Proceedings of the VLDB Endowment*, **1**(2), 1542–1552.

Douzal Chouakria A, Nagabhushan PN (2007). “Adaptive Dissimilarity Index for Measuring Time Series Proximity.” *Advances in Data Analysis and Classification*, **1**(1), 5–21.

D’Urso P, Maharaj EA (2009). “Autocorrelation-Based Fuzzy Clustering of Time Series.” *Fuzzy Sets and Systems*, **160**(24), 3565 – 3589.

Eiter T, Mannila H (1994). “Computing Discrete Fréchet Distance.” *Technical report*, Technische Universität Wien.

Fan J, Gijbels I (1996). *Local Polynomial Modelling and Its Applications*. Monographs on Statistics and Applied Probability, 66. Chapman & Hall.

Fan J, Kreutzberger E (1998). “Automatic Local Smoothing for Spectral Density Estimation.” *Scandinavian Journal of Statistics*, **25**(2), 359–369.

Fan J, Zhang W (2004). “Generalised Likelihood Ratio Tests for Spectral Density.” *Biometrika*, **91**(1), 195–209.

Fréchet MM (1906). “Sur Wuelques Points du Calcul Fonctionnel.” *Rendiconti del Circolo Matematico di Palermo (1884–1940)*, **22**(1), 1–72.

Fu TC (2011). “A Review on Time Series Data Mining.” *Engineering Applications of Artificial Intelligence*, **24**(1), 164–181.

Galeano P, Peña D (2000). “Multivariate Analysis in Vector Time Series.” *Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo*, **4**(4), 383–403.

Gavrilov M, Anguelov D, Indyk P, Motwani R (2000). “Mining the Stock Market: Which Measure is Best?” In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD’00, pp. 487–496. ACM.

Genolini C (2014). ***longitudinalData**: Longitudinal Data*. R package version 2.2, URL http://CRAN.R-project.org/package=longitudinalData.

Giorgino T (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The **dtw** Package.” *Journal of Statistical Software*, **31**(7), 1–24. URL http://www.jstatsoft.org/v31/i07/.

Golay X, Kollias S, Stoll G, Meier D, Valavanis A, Boesiger P (2005). “A New Correlation-Based Fuzzy Logic Clustering Algorithm for fMRI.” *Magnetic Resonance in Medicine*, **40**(2), 249–260.

Hennig C (2014). ***fpc**: Flexible Procedures for Clustering.* R package version 2.1-9, URL http://CRAN.R-project.org/package=fpc.

Kakizawa Y, Shumway RH, Taniguchi M (1998). "Discrimination and Clustering for Multivariate Time Series." *Journal of the American Statistical Association*, **93**(441), 328–340.

Kalpakis K, Gada D, Puttagunta V (2001). "Distance Measures for Effective Clustering of ARIMA Time-Series." In N Cercone, TY Lin, X Wu (eds.), *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 273–280.

Keogh E (2014). "The SAX (Symbolic Aggregate approXimation)." Accessed 2014-04-22, URL http://www.cs.ucr.edu/~eamonn/SAX.htm.

Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2000). "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases." *Journal of Knowledge and Information Systems*, **3**(3), 263–286.

Keogh E, Kasetty S (2003). "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration." *Data Mining and Knowledge Discovery*, **7**(4), 349–371.

Keogh E, Lonardi S, Ratanamahatana CA (2004). "Towards Parameter-Free Data Mining." In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'04, pp. 206–215. ACM, New York.

Keogh E, Lonardi S, Ratanamahatana CA, Wei L, Lee SH, Handley J (2007). "Compression-Based Data Mining of Sequential Data." *Data Mining and Knowledge Discovery*, **14**(1), 99–129.

Keogh E, Zhu Q, Hu B, Hao Y, Xi X, Wei L, Ratanamahatana CA (2011). "The UCR Time Series Classification/Clustering Homepage." URL http://www.cs.ucr.edu/~eamonn/time_series_data/.

Kovačić ZJ (1998). "Classification of Time Series with Applications to the Leading Indicator Selection." In *Data Science, Classification, and Related Methods – Proceedings of the Fifth Conference of the International Federation of Classification Societies (IFCS-96), Kobe, Japan, March 27–30, 1996*, pp. 204–207. Springer-Verlag.

Li M, Badger JH, Chen X, Kwong S, Kearney P, Zhang H (2001). "An Information-Based Sequence Distance and Its Application to Whole Mitochondrial Genome Phylogeny." *Bioinformatics*, **17**(2), 149–154.

Li M, Chen X, Li X, Ma B, Vitányi PMB (2004). "The Similarity Metric." *IEEE Transactions on Information Theory*, **50**(12), 3250–3264.

Li M, Vitányi P (2007). *An Introduction to Kolmogorov Complexity and Its Applications*. Text and Monographs in Computer Science. Springer-Verlag.

Liao TW (2005). "Clustering of Time Series Data : A Survey." *Pattern Recognition*, **38**(11), 1857–1874.

Lin J, Keogh E, Lonardi S, Chiu B (2003). "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms." In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD'03, pp. 2–11. ACM, New York.

Lin J, Keogh E, Wei L, Lonardi S (2007). "Experiencing SAX: A Novel Symbolic Representation of Time Series." *Data Mining and Knowledge Discovery*, **15**(2), 107–144.

Lin J, Li Y (2009). "Finding Structural Similarity in Time Series Data Using Bag-of-Patterns Representation." In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, pp. 461–477. Springer-Verlag, Berlin. ISBN 978-3-642-02278-4.

Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2014). ***cluster**: Cluster Analysis Basics and Extensions.* R package version 1.15.3, URL http://CRAN.R-project.org/package=cluster.

Maharaj EA (1996). "A Significance Test for Classifying ARMA Models." *Journal of Statistical Computation and Simulation*, **54**(4), 305–331.

Maharaj EA (2000). "Clusters of Time Series." *Journal of Classification*, **17**(2), 297–314.

Maharaj EA (2002). "Comparison of Non-stationary Time Series in the Frequency Domain." *Computational Statistics & Data Analysis*, **40**(1), 131–141.

Nieweglowski L (2013). ***clv**: Cluster Validation Techniques.* R package version 0.3-2.1, URL http://CRAN.R-project.org/package=clv.

Oates T, Firoiu L, Cohen PR (1999). "Clustering Time Series with Hidden Markov Models and Dynamic Time Warping." In *Proceedings of the IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pp. 17–21.

Ojeda Cabrera JL (2012). ***locpol**: Kernel Local Polynomial Regression.* R package version 0.6-0, URL http://CRAN.R-project.org/package=locpol.

Percival DB, Walden AT (2006). *Wavelet Methods for Time Series Analysis.* Cambridge University Press.

Pértega S, Vilar JA (2010). "Comparing Several Parametric and Nonparametric Approaches to Time Series Clustering: A Simulation Study." *Journal of Classification*, **27**(3), 333–362.

Piccolo D (1990). "A Distance Measure for Classifying ARIMA Models." *Journal of Time Series Analysis*, **11**(2), 153–164.

Popivanov I, Miller RJ (2002). "Similarity Search over Time-Series Data Using Wavelets." In *Proceedings of the 18th International Conference on Data Engineering*, pp. 212–221.

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012). "Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping." In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pp. 262–270. ACM, New York.

Ramoni M, Sebastiani P, Cohen P (2002). "Bayesian Clustering by Dynamics." *Machine Learning*, **47**(1), 91–121.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Ruppert D, Sheather SJ, Wand MP (1995). "An Effective Bandwidth Selector for Local Least Squares Regression." *Journal of the American Statistical Association*, **90**(432), 1257–1270.

Sankoff D, Kruskal JB (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley.

Sculley D, Brodley CE (2006). "Compression and Machine Learning: A New Perspective on Feature Space Vectors." In *Proceedings of the Data Compression Conference*, DCC'06, pp. 332–332. IEEE Computer Society, Washington.

Shumway RH, Unger AN (1974). "Linear Discriminant Functions for Stationary Time Series." *Journal of the American Statistical Association*, **69**, 948–956.

Smyth P (1997). "Clustering Sequences with Hidden Markov Models." In *Advances in Neural Information Processing Systems*, pp. 648–654. MIT Press.

Struzik ZR, Siebes A (1999). "The Haar Wavelet in the Time Series Similarity Paradigm." In *Principles of Data Mining and Knowledge Discovery – Proceedings of the Third European Conference, PKDD-99, Prague, Czech Republic, September 15–18, 1999*, pp. 12–22. Springer-Verlag.

Tan PN, Steinbach M, Kumar V (2006). *Introduction to Data Mining*. Pearson Addison Wesley.

Tong H, Yeung I (2000). "On Tests for Self-Exciting Threshold Autoregressive-Type Non-Linearity in Partially Observed Time Series." *Journal of the Royal Statistical Society C*, **40**(1), 43–62.

Vilar JA, Alonso AM, Vilar JM (2010). "Non-Linear Time Series Clustering Based on Non-Parametric Forecast Densities." *Computational Statistics & Data Analysis*, **54**(11), 2850–2865.

Vilar JA, Pértega S (2004). "Discriminant and Cluster Analysis for Gaussian Stationary Processes: Local Linear Fitting Approach." *Journal of Nonparametric Statistics*, **16**(3-4), 443–462.

Wand M (2014). ***KernSmooth**: Functions for Kernel Smoothing for Wand & Jones (1995)*. R package version 2.23-13, URL http://CRAN.R-project.org/package=KernSmooth.

Yi BK, Faloutsos C (2000). "Fast Time Sequence Indexing for Arbitrary L_p Norms." In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB'00, pp. 385–394. Morgan Kaufmann Publishers, San Francisco.

Zhang H, Ho TB, Zhang Y, Lin MS (2006). "Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform." *Informatica*, **30**(3), 305–319.

Affiliation:

Pablo Montero, José A. Vilar
Research Group on Modeling, Optimization and Statistical Inference (MODES)
Department of Mathematics
Computer Science Faculty
University of A Coruña
15071 A Coruña, Spain
E-mail: p.montero.manso@udc.es, jose.vilarf@udc.es

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 62, Issue 1 *Submitted:* 2013-06-16
November 2014 *Accepted:* 2014-07-11