Neurocomputing 537 (2023) 210–235

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

Machine learning for multivariate time series with the R package *mlmts*

Ángel López-Oriona [a,*], José A. Vilar [a]

[a] Research Group MODES, Research Center for Information and Communication Technologies (CITIC), University of A Coruña, 15071 A Coruña, Spain

ARTICLE INFO

Article history:
Received 18 June 2022
Revised 5 December 2022
Accepted 26 February 2023
Available online 11 March 2023
Communicated by Zidong Wang

Keywords:
mlmts
Multivariate time series
Clustering
Classification
Outlier detection
Forecasting
R package

ABSTRACT

Time series data are ubiquitous nowadays. Whereas most of the literature on the topic deals with univariate time series, multivariate time series have typically received much less attention. However, the development of machine learning algorithms for the latter objects has substantially increased in recent years. The R package **mlmts** attempts to provide a set of widespread data mining techniques for multivariate series. Several functions allowing the execution of clustering, classification, outlier detection and forecasting methods, among others, are included in the package. **mlmts** also incorporates a collection of multivariate time series datasets often used to test the performance of new classification algorithms. The main characteristics of the package are described and its use is illustrated through various examples. Practitioners from a wide variety of fields could benefit from the general framework provided by **mlmts**.

1. Introduction

Time series databases are becoming omnipresent nowadays, emerging frequently in a wide variety of disciplines as machine learning, biology, geology, finance, psychology, chemistry, among many other areas. Although univariate time series (UTS) were the norm until recently, the advance of technology and storage capabilities of everyday devices have provoked a growing presence of multivariate time series (MTS). Typical examples of MTS are multi-channel EEG signals or temporal records of concentrations of certain air pollutants in a specific region. Particularly significant in the last two decades has been the explosion of works on pattern recognition techniques for time series. [1,2] provide extensive surveys on current time series data mining directions including a number of important problems such as representation and indexing of series, visualization tools, dissimilarity measures, clustering, classification and anomaly detection.

Although a range of techniques concerning machine learning of multidimensional series are available in the literature, the majority of research has mainly focused on clustering (MTSCLU) and classification (MTSCLA) tasks. The former techniques attempt to split a set of unlabelled MTS realizations into homogeneous groups so that similar series are located together in the same group and dissimilar MTS are placed in distinct groups. On the other hand, MTSCLA algorithms learn on a collection of MTS with class labels, thus constructing a set of rules used to accurately predict the class of unlabelled series.

A pivotal issue in cluster analysis consists of establishing a suitable distance measure between two objects. This choice is particularly complex when dealing with time series due its dynamical nature, i.e., data objects evolving in time. Several approaches to define dissimilarity between MTS have been proposed. Sometimes the interest lies in comparing geometric profiles of the series, for which standard distances between raw data (e.g., Euclidean distance) can yield satisfactory results. For instance, two natural extensions of the dynamic time warping (DTW) distance to the multidimensional setting were introduced in [3]. Other times, dissimilarity is understood in terms of how different are the generating processes, and distance is then measured by comparing matrices or vectors of extracted serial features, as autocorrelations, cross-correlations, spectral quantities, quantile-based estimates, wavelet-based features and so on [4–7]. Alternative approaches consist of assuming specific underlying models [8], or performing dissimilarity reduction techniques and defining a metric in the reduced space [9,10]. In short, there exists a broad range of dissimilarities to compare MTS and the selection of the appropriate measure depends mainly on the purpose of the grouping task. Once the metric has been chosen, the execution of a clustering procedure is straightforward by considering, for instance, standard versions of

* Corresponding author.
E-mail addresses: oriona38@hotmail.com, a.oriona@udc.es (Á. López-Oriona), jose.vilarf@udc.es (J.A. Vilar).

https://doi.org/10.1016/j.neucom.2023.02.048
0925-2312/

K-means or *K*-medoids algorithms. Overviews on time series clustering (MTSCLU in particular) are provided by [11,12].

A range of algorithms for MTSCLA are also available in the literature. A classical technique consists of considering a multivariate extension of DTW, alone or in combination with an alternative dissimilarity, and then performing the classification task by using a *k* nearest neighbours (*k*NN) classifier [13–15]. Approaches relying on extracted features have also been proposed for MTSCLA. Likewise the feature-based methods for clustering, these procedures replace each MTS in the original set by a vector or matrix of statistical quantities, and perform the classification task by considering the new objects and a traditional classifier [4,16,17]. Other algorithms for MTSCLA are based on dimensionality reduction [18–20], word extraction [21] and neural networks [22]. Many of the previous classification approaches have been assessed in MTS datasets included in the UEA multivariate time series classification archive [23], which provides a collection of 30 MTS databases from different fields.

Besides clustering and classification procedures, there are other interesting and challenging problems related to MTS data mining. Among them, anomaly detection constitutes an important problem that has been addressed in different works [24–26]. Some of them perform the outlier identification task by assigning an outlier score to each MTS according to a given criteria.

According to previous considerations, having available a software tool where a range of machine learning algorithms for MTS can be tested and compared is highly desirable. In particular, we focus on the open-source R programming language [27]. There are many R packages to model and forecast uni and multidimensional time series, but this is far from being the case for clustering and classification. The R packages addressing these tasks are specifically tailored to univariate time series and most of them focus on specific approaches. For instance, **dtw** package [28] includes a wide variety of algorithms and constraints for the computation and visualization of DTW alignments, which have been widely used for clustering and classification of temporal data. More recently, **dtwclust** [29] provides algorithms and optimizations based on the DTW distance and targeted at clustering time series. Thus, additional implementations of partitional, hierarchical, fuzzy, *k*-Shape and TADPole clustering and of several cluster validity indexes are also available in **dtwclust**. Package **pdc** [30] implements a clustering method for time series based on measuring divergence between permutation distributions of the series. A few packages allow to use a wider range of approaches. For example, **TSclust** [31] and **TSdist** [32] include a large set of well-established peer-reviewed time series dissimilarity measures, and all of them can be used to perform conventional clustering algorithms. Nevertheless, as mentioned, these libraries focus on unidimensional time series. Sometimes the clustering algorithm can be extended to the multidimensional context (see e.g. the PDC algorithm in **pdc**), but usually this is not the case. In sum, there is a notorious lack of R libraries to address clustering and classification of multidimensional time series data, which justifies our attempt of providing practitioners and researchers with an integrated tool allowing the execution of several machine learning algorithms for MTS. The R package **mlmts** [33] is the result of those efforts. The package is structured so that its main functions are related to the computation of distances between MTS, thus attaching a high degree of homogeneity. However, it is worth emphasizing that the use of **mlmts** is not limited to clustering, since the distance matrix usually returned as output can be employed for other alternative data mining tasks. In addition, the functions associated with feature-based approaches allow obtain the extracted features, thus enabling their use as input to different types of algorithms, including classification ones. All the methods included in **mlmts** have been carefully examined in the MTS data mining literature and are appropriately referenced throughout this article. Nevertheless, some procedures require specific conditions to ensure meaningful results. In this regard, users should analyse with detail what is the most suitable function for their specific problem.

An interesting feature of the package **mlmts** is that it contains 28 of the 30 MTS datasets included in the UEA multivariate time series classification archive [23]. On the one hand, this allows users to check the results that the provided functions attain over these datasets from an exploratory point of view. On the other hand, this makes **mlmts** the perfect spot to test new MTSCLA algorithms, since the databases in the UEA archive are considered a standard location to evaluate the performance of novel classification algorithms in a reproducible manner.

In summary, **mlmts** package intends to integrate an extensive assortment of data mining algorithms for multidimensional series. In this way, the user can compare their behaviour and identify useful methods in a given context. **mlmts** is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/mlmts/index.html.

The rest of the paper is structured as follows. The distance measures implemented in **mlmts** are presented in Section 2. Specifically, we give a brief description of each dissimilarity and provide the corresponding references. Section 3 contains a short explanation about the data included in the package. In Section 4, the functionality of **mlmts** is illustrated through several examples. The performance of different algorithms is studied by considering synthetic data and some datasets included in the UEA archive. Some concluding remarks are given in Section 5.

2. Main functions in mlmts

The package **mlmts** contains a representative collection of data mining procedures for MTS. As stated in Section 1, most of the methods proposed in the literature make use of a pairwise dissimilarity matrix between the available collection of MTS objects. For this reason, our first challenge was to implement a set of functions providing a range of distances between MTS. The corresponding outputs are useful themselves, but also as starting point of other functions to run different MTS data mining algorithms, including clustering, classification and anomaly detection. This section is devoted to provide brief explanations about the considered dissimilarity measures, which have been grouped into three categories, namely shape-based approaches (Section 2.1), feature-based approaches (Section 2.2), model-based approaches (Section 2.3) and methods relying on dimensionality reduction techniques (Section 2.4). In addition, we present also four specific machine learning tools (Section 2.5).

From now on we assume that $\boldsymbol{X}_T = \left[\boldsymbol{X}_{T,1}^\top, \ldots, \boldsymbol{X}_{T,d}^\top\right]$ and $\boldsymbol{Y}_T = \left[\boldsymbol{Y}_{T,1}^\top, \ldots, \boldsymbol{Y}_{T,d}^\top\right]$ are partial realizations from two d-variate, real-valued stochastic processes $\{\boldsymbol{X}_t, t \in \mathbb{Z}\} = \{(X_{t,1}, \ldots, X_{t,d}), t \in \mathbb{Z}\}$ and $\{\boldsymbol{Y}_t, t \in \mathbb{Z}\} = \{(Y_{t,1}, \ldots, Y_{t,d}), t \in \mathbb{Z}\}$, respectively. Therefore, for $i = 1, \ldots, d$, $\boldsymbol{X}_{T,i} = \left(X_1^i, \ldots, X_T^i\right)$ and $\boldsymbol{Y}_{T,i} = \left(Y_1^i, \ldots, Y_T^i\right)$ denote the univariate realizations from the *ith* component of $\{\boldsymbol{X}_t, t \in \mathbb{Z}\}$ and $\{\boldsymbol{Y}_t, t \in \mathbb{Z}\}$, respectively. Note that realizations of the same length T and number of dimensions d are initially required, although some of the methods introduced throughout this section can deal with series of different lengths.

Based on previous considerations, a MTS can be represented by means of a matrix whose number of rows is the series length and whose number of columns is the number of variables or dimensions, i.e., $\boldsymbol{X}_T, \boldsymbol{Y}_T \in \mathbb{R}^{T \times d}$.

2.1. Shape-based distances

Proximity between MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be assessed through conventional metrics comparing raw values at specific points of time. Some common distances are presented below.

Euclidean distance

The Euclidean distance between MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is defined as the sum of the dimension-wise Euclidean distances, which is

$$d_{EUCL}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sum_{i=1}^{d} \left\| \boldsymbol{X}_{T,i} - \boldsymbol{Y}_{T,i} \right\|, \tag{1}$$

where $\left\| \boldsymbol{X}_{T,i} - \boldsymbol{Y}_{T,i} \right\|^2 = \sum_{t=1}^{T} \left(X_t^i - Y_t^i \right)^2$ is the classical Euclidean distance between two vectors of the same length. The Euclidean distance evaluates proximity of the values observed at corresponding dimensions and points of time. This metric is very sensitive to signal transformations as shifting or time scaling.

Fréchet distance

This metric was introduced by [34] to measure proximity between continuous curves but it has been extensively used in the time series framework. We first introduce the Fréchet distance in the univariate setting. Let $\boldsymbol{X}_T^U = (X_1, \ldots, X_T)$ and $\boldsymbol{Y}_T^U = (Y_1, \ldots, Y_T)$ be two UTS and let M be the set of all possible sequences of m pairs preserving the observations order in the form

$$r = \left((X_{a_1}, Y_{b_1}), \ldots, (X_{a_m}, Y_{b_m}) \right), \tag{2}$$

with $a_i, b_j \in \{1, \ldots, T\}$ 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 $i \in \{1, \ldots, m-1\}$. The Fréchet distance between UTS $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$ is defined as

$$d_{UF}\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U \right) = \min_{r \in M} \left(\max_{i=1,\ldots,m} |X_{a_i} - Y_{b_i}| \right). \tag{3}$$

Note that the Fréchet distance does not treat the series as two sets of points, but it takes into account the ordering of observations. Furthermore, d_{UF} can be computed in series of different lengths.

The Fréchet distance between MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is defined as

$$d_F(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sum_{i=1}^{d} d_{UF}(\boldsymbol{X}_{T,i}, \boldsymbol{Y}_{T,i}). \tag{4}$$

The distance d_{UF} is implemented in **mlmts** by using the R package **TSclust** [35].

Dynamic time warping distances

The dynamic time warping (DTW) distance is one of the most well-known shape-based dissimilarities for time series [36] in both the univariate and the multivariate setting. In the univariate context, DTW distance is aimed to find a mapping r (as in (2)) between the series so that a specific distance measure between the coupled observations (X_{a_i}, Y_{b_i}) is minimized. The DTW distance between UTS $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$ is given by

$$d_{DTW}\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U \right) = \min_{r \in M} \left(\sum_{i=1}^{m} |X_{a_i} - Y_{b_i}| \right). \tag{5}$$

As the distance d_{UF}, d_{DTW} allows to detect similar shapes, even in the presence of signal transformations. However, as the majority of geometric dissimilarities, d_{DTW} ignores the temporal structure of the values as the proximity is based on the differences $|X_{a_i} - Y_{b_i}|$ regardless of the behaviour around these values.

There exist two main extensions of DTW distance to the multivariate setting [3]. The "independent" version, denoted by d_{DTW1}, computes the distance d_{DTW} between the corresponding pairs of UTS, i.e.,

$$d_{DTW1}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sum_{i=1}^{d} d_{DTW}(\boldsymbol{X}_{T,i}, \boldsymbol{Y}_{T,i}). \tag{6}$$

On the other hand, the "dependent" version, hereafter referred to as d_{DTW2}, forces all dimensions to warp identically, in a single warping matrix. Specifically, let M^* be the set of all possible sequences of m^* pairs preserving the observations order in the form

$$r^* = \left((\boldsymbol{X}_{:,a_1}, \boldsymbol{Y}_{:,b_1}), \ldots, (\boldsymbol{X}_{:,a_{m^*}}, \boldsymbol{Y}_{:,b_{m^*}}) \right), \tag{7}$$

where $\boldsymbol{X}_{:,a_i}$ and $\boldsymbol{Y}_{:,b_j}$ denote the a_ith row of $\boldsymbol{X}_T$ and the b_jth row of $\boldsymbol{Y}_T$, respectively, and the sequences indexed by a_i and b_j are subject to the same conditions as in (2).

The distance d_{DTW2} is defined as

$$d_{DTW2}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \min_{r^* \in M^*} \left(\sum_{i=1}^{m^*} \left\| \boldsymbol{X}_{:,a_i} - \boldsymbol{Y}_{:,b_i} \right\| \right). \tag{8}$$

Both d_{DTW1} and d_{DTW2} have proven successful in several MTS data mining tasks [37,38]. In **mlmts**, the distances d_{DTW1} and d_{DTW2} are implemented with the help of the R package **dtw** [29].

Mahalanobis distance

The Mahalanobis distance is a classical metric in multivariate data analysis. This measure has been extensively employed in the context of MTS data mining [39,10,40]. Define $\overline{\boldsymbol{X}}_T = (\overline{X}_1, \ldots, \overline{X}_d)$ and $\overline{\boldsymbol{Y}}_T = (\overline{Y}_1, \ldots, \overline{Y}_d)$, where

$$\overline{X}_j = \frac{1}{T} \sum_{t=1}^{T} X_t^j \quad \text{and} \quad \overline{Y}_j = \frac{1}{T} \sum_{t=1}^{T} Y_t^j, \tag{9}$$

are the mean levels of UTS $\boldsymbol{X}_{T,j}$ and $\boldsymbol{Y}_{T,j}$, respectively. The Mahalanobis divergence between MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is given by [10]

$$d_{MD}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sqrt{(\overline{\boldsymbol{X}}_T - \overline{\boldsymbol{Y}}_T) \boldsymbol{\Sigma}_{\boldsymbol{X}_T}^{*-1} (\overline{\boldsymbol{X}}_T - \overline{\boldsymbol{Y}}_T)^\top}, \tag{10}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{X}_T}$ is the covariance matrix of $\boldsymbol{X}_T$ and $\boldsymbol{\Sigma}_{\boldsymbol{X}_T}^{*-1}$ is the pseudo-inverse of $\boldsymbol{\Sigma}_{\boldsymbol{X}_T}$ calculated using Singular Value Decomposition (SVD). In (10), MTS $\boldsymbol{X}_T$ is assumed to be the reference series.

Note that d_{MD} is not a distance since it does not satisfy the symmetric property. In **mlmts**, a symmetric version of d_{MD} is implemented by defining

$$d_{MD}^*(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \frac{1}{2} \left(d_{MD}(\boldsymbol{X}_T, \boldsymbol{Y}_T) + d_{MD}(\boldsymbol{Y}_T, \boldsymbol{X}_T) \right). \tag{11}$$

As the previously introduced metrics, d_M^* pertains to the class of shape-based dissimilarities. However, unlike the former distances, d_M^* does not consider point to point examinations, but an overall comparison between levels. In addition, d_M^* accounts for possible correlations among the d dimensions.

A distance measure relying on both a DTW-type and a Mahalanobis-type dissimilarity is described below.

Mahalanobis distance based dynamic time warping

A Mahalanobis distance based on dynamic time warping (MDDTW) was proposed in [14] for MTS classification. Here, we adapt this metric to a general setting.

Preserving the notation of (7), the MDDTW distance between two MTS is expressed as

$$d_{MDDTW}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \min_{r^* \in M^*} \sum_{i=1}^{m^*} \frac{1}{2} (A_i + B_i), \tag{12}$$

with A_i and B_i are defined by $A_i = (\boldsymbol{X}_{:,a_i} - \boldsymbol{Y}_{:,b_i}) \boldsymbol{\Sigma}_{\boldsymbol{X}_T}^{*-1} (\boldsymbol{X}_{:,a_i} - \boldsymbol{Y}_{:,b_i})^\top$ and $B_i = (\boldsymbol{X}_{:,a_i} - \boldsymbol{Y}_{:,b_i}) \boldsymbol{\Sigma}_{\boldsymbol{Y}_T}^{*-1} (\boldsymbol{X}_{:,a_i} - \boldsymbol{Y}_{:,b_i})^\top$.

Note that the metric d_{MDDTW} is an extension of d_{DTW2}, where the local distance between the row vectors of both MTS is computed according to a squared Mahalanobis-type dissimilarity.

2.2. Feature-based distances

Several dissimilarity measures based on feature extraction are presented in this section. These techniques rely on two steps: (i) a collection of statistical quantities are computed from each MTS and (ii) the distance between two series is obtained by comparing the corresponding sets of features. It is worth highlighting that, although most feature extraction methods are generic in nature, the extracted characteristics are usually application dependent.

Correlation-based distances

[41] proposed to assess dissimilarity between UTS by using proper estimates of the autocorrelation function up to some prefixed lag. Here we extend their approach to the multivariate setting.

Fix $l \in \mathbb{Z}$ and let $\widehat{\rho}^{\boldsymbol{X}_T}_{j_1 j_2}(l) = \widehat{\text{Cor}}(X_{tj_1}, X_{t+lj_2})$ and $\widehat{\rho}^{\boldsymbol{Y}_T}_{j_1 j_2}(l) = \widehat{\text{Cor}}(Y_{tj_1}, Y_{t+lj_2})$, for $1 \leqslant j_1, j_2 \leqslant d$, be the estimated cross-correlations for lag l computed from the realizations $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively.

A metric comparing estimates of cross-correlations up to lag L can be defined as

$$d_{COR}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left[\sum_{l=1}^{L}\sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\left(\widehat{\rho}^{\boldsymbol{X}_T}_{j_1 j_2}(l) - \widehat{\rho}^{\boldsymbol{Y}_T}_{j_1 j_2}(l)\right)^2 + \sum_{\substack{j_1 j_2 = 1: \\ j_1 > j_2}}^{d}\left(\widehat{\rho}^{\boldsymbol{X}_T}_{j_1 j_2}(0) - \widehat{\rho}^{\boldsymbol{Y}_T}_{j_1 j_2}(0)\right)^2\right]^{1/2}. \tag{13}$$

Note that the second term of (13) involves the lag $l = 0$ to assess simultaneous relationships between the single components of each MTS, fixing here $j_1 > j_2$ due to the symmetric character of the cross-correlation function when the lag is zero.

[42] constructed a measure of linear dependence between UTS, which is employed to perform hierarchical clustering of UTS. An extension of this metric to the multivariate context is available in the package **mlmts**.

Let $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$ be two stationary UTS of length T, i.e., two realizations from the univariate stochastic process $\{X_t, t \in \mathbb{Z}\}$ and $\{Y_t, t \in \mathbb{Z}\}$, respectively. Without loss of generality, it is assumed that $E(X_t) = E(Y_t) = 0$ and $E\left(X_t^2\right) = E\left(Y_t^2\right) = 1$. Define $\rho_X(l) = E(X_{t-l}X_t)$, $\rho_Y(l) = E(Y_{t-l}Y_t)$ and $\rho_{XY}(l) = E(X_{t-l}Y_t)$.

The linear dependence for lags between 0 and L can be summarized by means of the matrix.

$$\boldsymbol{R}_{YX,L} = \begin{pmatrix} 1 & \rho_Y(1) & \cdots & \rho_Y(L) & \rho_{XY}(0) & \rho_{XY}(1) & \cdots & \rho_{XY}(L) \\ \rho_Y(1) & 1 & \cdots & \rho_Y(L-1) & \rho_{XY}(-1) & \rho_{XY}(0) & \cdots & \rho_{XY}(L-1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_Y(L) & \rho_Y(L-1) & \cdots & 1 & \rho_{XY}(-L) & \rho_{XY}(-L+1) & \cdots & \rho_{XY}(0) \\ \rho_{XY}(0) & \rho_{XY}(-1) & \cdots & \rho_{XY}(-L) & 1 & \rho_X(1) & \cdots & \rho_X(L) \\ \rho_{XY}(1) & \rho_{XY}(0) & \cdots & \rho_{XY}(-L+1) & \rho_X(1) & 1 & \cdots & \rho_X(L-1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{XY}(L) & \rho_{XY}(L-1) & \cdots & \rho_{XY}(0) & \rho_X(L) & \rho_X(L-1) & \cdots & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_{YY,L} & \boldsymbol{C}_{XY,L}^\top \\ \boldsymbol{C}_{XY,L} & \boldsymbol{R}_{XX,L} \end{pmatrix}, \tag{14}$$

where $\boldsymbol{R}_{XX,L}$ is the covariance matrix of $(X_t, X_{t-1}, \ldots, X_{t-L})^\top$, $\boldsymbol{R}_{YY,L}$ is the covariance matrix of $(Y_t, Y_{t-1}, \ldots, Y_{t-L})^\top$ and $\boldsymbol{C}_{XY,L}$ includes the cross-correlations between both vector processes. Matrix $\boldsymbol{R}_{YX,L}$ is the covariance matrix of the vector stationary process $(Y_t, Y_{t-1}, \ldots, Y_{t-L}, X_t, X_{t-1}, \ldots, X_{t-L})^\top$.

The so-called generalized cross-correlation between UTS $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$ is defined as

$$GCC\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U\right) = 1 - \left(\frac{|\widehat{\boldsymbol{R}}_{YX,L}|}{|\widehat{\boldsymbol{R}}_{XX,L}||\widehat{\boldsymbol{R}}_{YY,L}|}\right)^{1/(L+1)}, \tag{15}$$

where the estimators $\widehat{\boldsymbol{R}}_{YX,L}$, $\widehat{\boldsymbol{R}}_{XX,L}$ and $\widehat{\boldsymbol{R}}_{YY,L}$ are obtained by considering the classical estimators of auto and cross-correlations in the matrix $\boldsymbol{R}_{YX,L}$. The measure *GCC* verifies: (i) $GCC\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U\right) = GCC\left(\boldsymbol{Y}_T^U, \boldsymbol{X}_T^U\right)$, (ii) $GCC\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U\right) \in [0,1]$, (iii) $GCC\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U\right) = 1$ if and only if there is a perfect linear dependence between the series and $GCC\left(\boldsymbol{X}_T^U, \boldsymbol{Y}_T^U\right) = 0$ if and only if all the cross-correlation coefficients are zero. Therefore, *GCC* provides a great deal of information about the linear dependence between $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$.

A distance measure between MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be established by comparing the pair-wise linear relationships exhibited by each set of dimensions in terms of the quantity *GCC*. Specifically, the metric is given by

$$d_{GCC}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left[\sum_{j_1=1}^{d}\sum_{\substack{j_2=1: \\ j_1 > j_2}}^{d}\left(GCC(\boldsymbol{X}_{Tj_1}, \boldsymbol{X}_{Tj_2}) - GCC(\boldsymbol{Y}_{Tj_1}, \boldsymbol{Y}_{Tj_2})\right)^2\right]^{1/2}. \tag{16}$$

Spectral-based distances

Two dissimilarity measures based on estimated spectral quantities were proposed by [4] to perform machine learning procedures in a set of multivariate series.

Let $\Omega = \{\omega_k, k = 1, \ldots, K\}$ be the set of K Fourier frequencies and $\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}(\omega_k)$ and $\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}(\omega_k)$ be the spectral matrices estimators for series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, computed as the usual nonparametric estimators. A measure of discrepancy based on the so-called J divergence is defined as

$$d_{JSPEC}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \frac{1}{2T}\sum_{k=1}^{K}\left(\text{tr}\left(\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}(\omega_k)\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}^{-1}(\omega_k)\right) + \text{tr}\left(\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}(\omega_k)\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}^{-1}(\omega_k)\right) - 2d\right), \tag{17}$$

where $\text{tr}(\cdot)$ denotes the trace of a square matrix.

A dissimilarity measure relying on the so-called Chernoff information divergence is given by

$$d_{CSPEC}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \frac{1}{2T}\sum_{k=1}^{K}\left(\log\frac{|\alpha\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}(\omega_k) + (1-\alpha)\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}(\omega_k)|}{|\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}(\omega_k)|} + \log\frac{|\alpha\widehat{\boldsymbol{f}}_{\boldsymbol{Y}_T}(\omega_k) + (1-\alpha)\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}(\omega_k)|}{|\widehat{\boldsymbol{f}}_{\boldsymbol{X}_T}(\omega_k)|}\right), \tag{18}$$

where $\alpha \in (0,1)$. The quantity d_{CSPEC} tends to behave like a Kullback–Leibler measure for values of the parameter α that are near the boundaries 0 and 1.

It is worth remarking that both d_{JSPEC} and d_{CSPEC} are not real distance measures, since they do not satisfy the triangle inequality. Both quantities were used in [4] to construct hierarchical and partitioning clustering techniques, as well as classification algorithms based on discriminant analysis. In addition, the measure d_{JSPEC} was broadly assessed in [7] in a clustering context, achieving excellent results when grouping linear series.

The nonparametric estimators $\widehat{\boldsymbol{f}}^{\boldsymbol{X}_T}(\omega_k)$ and $\widehat{\boldsymbol{f}}^{\boldsymbol{Y}_T}(\omega_k)$ in expressions (17) and (18) are computed in **mlmts** using the package **freqdom** [43].

Wavelet-based distance

The maximum overlap discrete wavelet transform (MODWT) was used by [6] to construct crisp and fuzzy clustering methods for MTS. The MODWT is a modification of the discrete wavelet

transform (DWT) ensuring that the MODWT wavelet coefficients at each scale have the same length as the original time series, thus overcoming the lack of time invariance of the latter (see Section 2 in [6]).

Let $\{X_t, t \in \mathbb{Z}\}$ be a univariate stochastic process. Denote by $\{h_{jl} : l = 0, \ldots, L_j\}$ the MODWT wavelet filter of length L_j associated with the scale $v_j = 2^{j-1}$. Then, define the MODWT wavelet coefficients of level j by means the convolution of the time series and the MODWT filters, i.e., by considering $W_{X,jt} = \sum_{l=0}^{L_j} h_{jl} X_{t-l}$. If it exists and is finite, the time independent variance at scale v_j is defined as $v_X^2(v_j) = \mathrm{Var}(W_{X,jt})$ and the equality $\sum_{j=1}^{\infty} v_X^2(v_j) = \mathrm{Var}(X_t)$ holds.

Given a UTS $\boldsymbol{X}_T^U$, which is a realization of the stochastic process $\{X_t, t \in \mathbb{Z}\}$, an unbiased estimator of $v_X^2(v_j)$ can be obtained by means of

$$\widehat{v}_X^2(v_j) = \frac{1}{M_j} \sum_{t=L_j}^{T-1} \widehat{W}_{X,jt}^2, \tag{19}$$

where $\widehat{W}_{X,jt}^2$ are MODWT coefficients associated with the time series $\boldsymbol{X}_T^U$ and $M_j = T - L_j + 1$ are the number of wavelet coefficients excluding the boundary coefficients that are affected by the circular assumption of the wavelet filter. Let $\{Y_t, t \in \mathbb{Z}\}$, be another univariate stochastic processes with MODWT coefficients $W_{Y,jt}$. The wavelet covariance can be defined as $v_{XY}(v_j) = \mathrm{Cov}(W_{X,jt}, W_{Y,jt})$, giving a scale-based decomposition of the covariance between X_t and Y_t, i.e., $\sum_{j=1}^{\infty} v_{XY}(v_j) = \mathrm{Cov}(X_t, Y_t)$.

Similarly, the wavelet correlation at scale v_j is defined as $\psi_{XY}(v_j) = \frac{v_{XY}(v_j)}{v_X^2(v_j) v_Y^2(v_j)}$. For two univariate series $\boldsymbol{X}_T^U$ and $\boldsymbol{Y}_T^U$, respectively, the estimator of $\psi_{XY}(v_j)$ is obtained by replacing $v_{XY}(v_j)$, $v_X^2(v_j)$ and $v_Y^2(v_j)$ by their estimators. Thus, by considering unbiased estimators $\widehat{v}_{\boldsymbol{X}_T^U \boldsymbol{Y}_T^U}(v_j)$, $\widehat{v}_{\boldsymbol{X}_T^U}^2(v_j)$ and $\widehat{v}_{\boldsymbol{Y}_T^U}^2(v_j)$, we obtain $\widehat{\psi}_{\boldsymbol{X}_T^U \boldsymbol{Y}_T^U}(v_j) = \frac{\widehat{v}_{\boldsymbol{X}_T^U \boldsymbol{Y}_T^U}(v_j)}{\widehat{v}_{\boldsymbol{X}_T^U}^2(v_j) \widehat{v}_{\boldsymbol{Y}_T^U}^2(v_j)}$. A measure of discrepancy between the multivariate series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be constructed by comparing estimates of their wavelet variances and wavelet correlations for the different components of the multivariate series, i.e., by considering the metric

$$d_{MODWT}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left[\sum_{i=1}^{J} \sum_{j=1}^{d} \left(\widehat{v}_{\boldsymbol{X}_{T,j}}^2(v_i) - \widehat{v}_{\boldsymbol{Y}_{T,j}}^2(v_i) \right)^2 + \sum_{i=1}^{J} \sum_{\substack{j_1, j_2 = 1: \\ j_1 > j_2}}^{d} \left(\widehat{\psi}_{\boldsymbol{X}_{T,j_1} \boldsymbol{X}_{T,j_2}}(v_i) - \widehat{\psi}_{\boldsymbol{Y}_{T,j_1} \boldsymbol{Y}_{T,j_2}}(v_i) \right)^2 \right]^{1/2}, \tag{20}$$

where J is the maximum allowable number of scales. Note that the summation in the second term of (20) imposes that $j_1 > j_2$ due to the symmetric character of the matrix of the wavelet correlation.

The computation of estimated wavelet variances and correlations is carried out in **mlmts** by means of the package **waveslim** [44]. Several wavelet filters are available to the user in order to calculate the corresponding estimates.

Quantile-based distances

A distance measure based on comparing quantile autocovariance functions is introduced in [45] to perform clustering of UTS. The package **mlmts** implements an extension of this metric which is able to deal with MTS objects.

Let $\{\boldsymbol{X}_t, t \in \mathbb{Z}\} = \{(X_{t,1}, \ldots, X_{t,d}), t \in \mathbb{Z}\}$ be a d-variate real-valued strictly stationary stochastic process. Denote by F_j the marginal distribution function of $X_{t,j}, j = 1, \ldots, d$, and by $q_j(\tau) = F_j^{-1}(\tau), \tau \in [0, 1]$, the corresponding quantile function. Fixed $l \in \mathbb{Z}$ and an arbitrary couple of quantile levels $(\tau, \tau') \in [0, 1]^2$, consider the cross-covariance of the indicator functions $I\{X_{t,j_1} \leqslant q_{j_1}(\tau)\}$ and $I\{X_{t+l,j_2} \leqslant q_{j_2}(\tau')\}$ given by

$$\gamma_{j_1 j_2}(l, \tau, \tau') = \mathrm{Cov}\left(I\{X_{t,j_1} \leqslant q_{j_1}(\tau)\}, I\{X_{t+l,j_2} \leqslant q_{j_2}(\tau')\} \right), \tag{21}$$

for $1 \leqslant j_1, j_2 \leqslant d$. Quantity (21) is called the quantile cross-covariance. It is always well-defined (even for processes with infinite moments) and examines the sequential dependence structure of the series that the standard cross-covariance is not able to capture. Furthermore, quantile cross-covariance takes advantage from the local distributional properties inherent to the quantile methods and it can be easily estimated by replacing the theoretical quantiles by the empirical quantiles, thus obtaining the estimate $\widehat{\gamma}_{j_1 j_2}(l, \tau, \tau')$. A simple dissimilarity criterion between a pair of MTS consists of comparing their estimated quantile cross-covariances up to lag L evaluated on a common range of selected quantiles and for every couple of dimensions, i.e.,

$$d_{QCF}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left[\sum_{l=1}^{L} \sum_{i=1}^{r} \sum_{i'=1}^{r} \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \left(\widehat{\gamma}_{j_1 j_2}^{\boldsymbol{X}_T}(l, \tau_i, \tau_{i'}) - \widehat{\gamma}_{j_1 j_2}^{\boldsymbol{Y}_T}(l, \tau_i, \tau_{i'}) \right)^2 + \sum_{i=1}^{r} \sum_{i'=1}^{r} \sum_{\substack{j_1, j_2 = 1: \\ j_1 > j_2}}^{d} \left(\widehat{\gamma}_{j_1 j_2}^{\boldsymbol{X}_T}(0, \tau_i, \tau_{i'}) - \widehat{\gamma}_{j_1 j_2}^{\boldsymbol{Y}_T}(0, \tau_i, \tau_{i'}) \right)^2 \right]^{1/2}, \tag{22}$$

where the superscripts $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ indicate that the estimators $\widehat{\gamma}_{j_1 j_2}(l, \tau_i, \tau_{i'})$ are computed according to the realizations $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, and $\mathscr{T} = \{\tau_1, \ldots, \tau_r\}$ is a set of r probability levels.

The distance d_{QCF} is a quantile-based metric in the time domain. Now we introduce its counterpart in the frequency domain. Under suitable summability conditions (mixing conditions), the Fourier transform of $\gamma_{j_1 j_2}(l, \tau, \tau')$ is well-defined and the quantile cross-spectral density is given by

$$\mathfrak{f}_{j_1 j_2}(\omega, \tau, \tau') = (1/2\pi) \sum_{l=-\infty}^{\infty} \gamma_{j_1 j_2}(l, \tau, \tau') e^{-il\omega}, \tag{23}$$

where $\omega \in \mathbb{R}$. The quantile cross-spectral density can be estimated consistently by means of the so-called smoothed CCR-periodogram proposed by [46], denoted by, $\widehat{G}_{j_1 j_2}(\omega, \tau, \tau')$. Let $\Omega = \{\omega_k, k = 1, \ldots, K\}$ be the set of K Fourier frequencies. A distance measure between series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be established by comparing their representations in terms of smoothed CCR-periodograms evaluated in the sets $\Omega, \mathscr{T}$ and every couple of dimensions [7], obtaining

$$d_{QCD}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left[\sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \sum_{i=1}^{r} \sum_{i'=1}^{r} \sum_{k=1}^{K} \left(\Re\left(\widehat{G}_{j_1 j_2}^{\boldsymbol{X}_T}(\omega_k, \tau_i, \tau_{i'}) \right) - \Re\left(\widehat{G}_{j_1 j_2}^{\boldsymbol{Y}_T}(\omega_k, \tau_i, \tau_{i'}) \right) \right)^2 + \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \sum_{i=1}^{r} \sum_{i'=1}^{r} \sum_{k=1}^{K} \left(\Im\left(\widehat{G}_{j_1 j_2}^{\boldsymbol{X}_T}(\omega_k, \tau_i, \tau_{i'}) \right) - \Im\left(\widehat{G}_{j_1 j_2}^{\boldsymbol{Y}_T}(\omega_k, \tau_i, \tau_{i'}) \right) \right)^2 \right]^{1/2}, \tag{24}$$

where the superscripts $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ indicate that the estimators $\widehat{G}_{j_1 j_2}(\omega_k, \tau_i, \tau_{i'})$ are computed according to the realizations $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively, $\Re$ denotes the real part and $\Im$ denotes the imaginary part of a complex number. The smoothed CCR-periodograms appearing in the expression of d_{QCD} are computed in **mlmts** by using the package **quantspec** [47].

Both dissimilarities d_{QAF} and d_{QCD} have been successfully employed in the context of unsupervised learning. Specifically, they proved very effective in grouping time series generated from a wide variety of underlying stochastic processes.

Dissimilarity measures based on several statistical quantities

Some approaches for MTS data mining rely on describing each series by means of several statistical quantities of different nature. The computed feature vectors are used to perform a concrete machine learning task, as clustering, classification or outlier detection, among others. Three approaches based on the idea of versatile feature extraction are available in **mlmts**. Particularly, the techniques described in [5,25,16] are implemented. The specific extracted quantities are not described in this manuscript for the sake of simplicity so that the reader is referred to the corresponding references for more details.

Let $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ be two multivariate series and $\widehat{\boldsymbol{\theta}}_k^{\boldsymbol{X}_T}$ and $\widehat{\boldsymbol{\theta}}_k^{\boldsymbol{Y}_T}, k \in \{WWW, HWL, Z\}$ be the vectors of extracted quantities associated with MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ according to the procedures described in [5,25,16], respectively. According to prior considerations, three dissimilarity measures between MTS defined in the space of feature vectors are given by

$$d_{WWW}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left\| \widehat{\boldsymbol{\theta}}_{WWW}^{\boldsymbol{X}_T} - \widehat{\boldsymbol{\theta}}_{WWW}^{\boldsymbol{Y}_T} \right\|, \; d_{HWL}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left\| \widehat{\boldsymbol{\theta}}_{HWL}^{\boldsymbol{X}_T} - \widehat{\boldsymbol{\theta}}_{HWL}^{\boldsymbol{Y}_T} \right\|,$$
$$d_{ZAGORECKI}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \left\| \widehat{\boldsymbol{\theta}}_{ZAGORECKI}^{\boldsymbol{X}_T} - \widehat{\boldsymbol{\theta}}_{ZAGORECKI}^{\boldsymbol{Y}_T} \right\|. \tag{25}$$

2.3. Model-based distances

Model-based metrics assume that the time series are generated from specific underlying models. The few works available in the literature suppose that the MTS follow a Vector Autoregressive Moving Average (VARMA) model. In such case, the idea is fitting a VARMA model to each series and then assessing discrepancy between the fitted models. The structure is automatically determined by using, for instance, the Akaike's information criterion (AIC) or the Schwarz's Bayesian information criterion (BIC). The parameter estimates are computed by means of generalized least squares estimators. Two important distance measures employing the assumption of underlying VARMA models are described in this section.

A distance based on estimated VAR coefficients

In the univariate setting, [48] introduced a dissimilarity measure for the class of invertible ARIMA processes, namely the Euclidean distance between the estimated coefficients of the truncated AR(∞) structures approximating the underlying ARIMA models. The package **mlmts** contains a generalization of Piccolo's distance to the multivariate setting. First, a specific criterion such as AIC or BIC is employed to fit VAR models of orders k_1 and k_2 to the multivariate series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively. Then, the Euclidean distance between the corresponding vectorized versions of parameter estimates is computed, where proper zero-padding is used when $k_1 \neq k_2$.

Let $\widehat{\boldsymbol{\Gamma}}_{\boldsymbol{X}_T} = \left\{ \widehat{\boldsymbol{\Pi}}_{\boldsymbol{X}_T}^1, \ldots, \widehat{\boldsymbol{\Pi}}_{\boldsymbol{X}_T}^{k_1}, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}_T}^{\epsilon} \right\}$ and $\widehat{\boldsymbol{\Gamma}}_{\boldsymbol{Y}_T} = \left\{ \widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^1, \ldots, \widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^{k_2}, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}_T}^{\epsilon} \right\}$ be the sets of estimated matrices of coefficients for series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively. Specifically, $\widehat{\boldsymbol{\Pi}}_{\boldsymbol{X}_T}^j$ ($\widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^j$) denotes the estimated matrix of coefficients associated with lag j for $\boldsymbol{X}_T$ ($\boldsymbol{Y}_T$), and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}_T}^{\epsilon}$ ($\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}_T}^{\epsilon}$) denotes the estimated covariance matrix of the error process for $\boldsymbol{X}_T$ ($\boldsymbol{Y}_T$), $j = 1, \ldots, k_1$ ($j = 1, \ldots, k_2$). Note that both $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ were assumed to be zero-mean. Without loss of generality, suppose that $k_1 \geqslant k_2$ and define the padded counterpart of $\widehat{\boldsymbol{\Gamma}}_{\boldsymbol{Y}_T}$ as

$$\widehat{\boldsymbol{\Gamma}}_{\boldsymbol{Y}_T}^p = \begin{cases} \left\{ \widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^1, \ldots, \widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^{k_2}, \boldsymbol{0}_{d\times d}^{k_2+1}, \ldots, \boldsymbol{0}_{d\times d}^{k_2+k_1}, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}_T}^{\epsilon} \right\}, & k_1 > k_2, \\ \widehat{\boldsymbol{\Gamma}}_{\boldsymbol{Y}_T}, & k_1 = k_2, \end{cases} \tag{26}$$

where $\boldsymbol{0}_{d\times d}^j \in \mathbb{R}^{d\times d}$ is a matrix with all the entries equal to zero, $j = k_2, \ldots, k_2 + k_1$. The generalized Piccolo's distance takes the form

$$d_{VAR}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = ||\left(vec\left(\widehat{\boldsymbol{\Pi}}_{\boldsymbol{X}_T}^1\right), \ldots, vec\left(\widehat{\boldsymbol{\Pi}}_{\boldsymbol{X}_T}^{k_1}\right), vec\left(\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}_T}^{\epsilon}\right)\right) - \left(vec\left(\widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^1\right), \ldots, vec\left(\widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^{k_2}\right), \ldots, vec\left(\widehat{\boldsymbol{\Pi}}_{\boldsymbol{Y}_T}^{k_2+k_1}\right), vec\left(\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}_T}^{\epsilon}\right)\right)||, \tag{27}$$

where the operator $vec(\cdot)$ creates a vector by concatenating the columns of the matrix received as input. Note that, unlike the original Piccolo's distance [48], the generalized Piccolo's distance takes into account the estimates of the variability of the error process.

The metric d_{VAR} has been extensively analysed in a clustering framework by [7]. Specifically, its behaviour was studied in a broad simulation study including several types of processes. As expected, d_{VAR} achieved great results when dealing with linear processes. However, when the underlying models deviated from the linearity assumption, the distance exhibited very poor performance, failing to detect the underlying groups. In sum, the dissimilarity d_{VAR} is restricted to a limited number of scenarios, since a great percentage of real MTS datasets are known to display nonlinear patterns.

A distance based on a hypothesis test

In an early work, [8] introduced a distance metric based on a test of hypothesis to determine whether or not two stationary MTS have been generated from different stochastic processes. In fact, the dissimilarity measure is based on the p-value of the corresponding test. First, truncated VAR(∞) models of order k are fitted to each series. Afterwards, a test statistic assessing differences between the VAR(k) estimates of both series is computed, and the associated p-value is obtained from the corresponding asymptotic distribution of the test statistic under the null of equal generating processes. A nice property of the proposed hypothesis test is that it can be applied to time series that are not necessarily independent. Below we provide a brief description of the procedure.

Assume that $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ are stationary MTS which have been generated from VAR(∞) processes. Using a definite criterion such as AIC or BIC, truncated VAR(∞) of order k_1 and k_2 are fitted to $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively. Suppose without loss of generality that $k_1 = k_2 = k$ (otherwise we proceed as indicated when defining the distance d_{VAR}). Define the parameter matrices of the generating processes $\{\boldsymbol{X}_t, t \in \mathbb{Z}\}$ and $\{\boldsymbol{Y}_t, t \in \mathbb{Z}\}$, respectively, as $\boldsymbol{\Pi}_{\boldsymbol{X}_t} = \left[\boldsymbol{\Pi}_{\boldsymbol{X}_t}^1, \ldots, \boldsymbol{\Pi}_{\boldsymbol{X}_t}^k\right]$ and $\boldsymbol{\Pi}_{\boldsymbol{Y}_t} = \left[\boldsymbol{\Pi}_{\boldsymbol{Y}_t}^1, \ldots, \boldsymbol{\Pi}_{\boldsymbol{Y}_t}^k\right]$, with $\boldsymbol{\Pi}_{\boldsymbol{X}_t}, \boldsymbol{\Pi}_{\boldsymbol{Y}_t} \in \mathbb{R}^{d\times dk}$. The hypotheses to be tested are:

H_0: There is no difference between the generating process of both multivariate series, i.e., $vec(\boldsymbol{\Pi}_{\boldsymbol{X}_t}) = vec(\boldsymbol{\Pi}_{\boldsymbol{Y}_t})$.

H_1: There is a difference between the generating process of both multivariate series, i.e., $vec(\boldsymbol{\Pi}_{\boldsymbol{X}_t}) \neq vec(\boldsymbol{\Pi}_{\boldsymbol{Y}_t})$.

The $d(T-k)$ equations fitted to $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be expressed collectively as the following regression model:

$$\boldsymbol{Z}_v^\top = \boldsymbol{B}_v \boldsymbol{\Pi}_v^\top + \boldsymbol{A}_v^\top, \tag{28}$$

where $\boldsymbol{\Pi}_v = \begin{bmatrix} vec(\boldsymbol{\Pi}_{\boldsymbol{X}_t})^\top \\ vec(\boldsymbol{\Pi}_{\boldsymbol{Y}_t})^\top \end{bmatrix}^\top$ is the vector of parameters, $\boldsymbol{Z}_v$ is the response vector, $\boldsymbol{B}_v$ is the matrix of predictors, and $\boldsymbol{A}_v$ is the error vector. Both $\boldsymbol{Z}_v$ and $\boldsymbol{B}_v$ are totally constructed from the series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ (see Section 2 in [8] for more details). Note that the null hypothesis can be expressed as H_0: $\boldsymbol{R}\boldsymbol{\Pi}_v^\top = \boldsymbol{0}_{d^2k\times 1}$, with $\boldsymbol{R} = \left[i_{d^2k\times d^2k}, -i_{d^2k\times d^2k}\right]$, where $i_{d^2k\times d^2k} \in \mathbb{R}^{d^2k\times d^2k}$ is the identity matrix. Let $\boldsymbol{\Sigma}_{\boldsymbol{A}_v} = Cov(\boldsymbol{A}_v)$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{A}_v}$ be the ordinary least squares estimator of $\boldsymbol{\Sigma}_{\boldsymbol{A}_v}$ according to the regression model (28). In this way, the feasible least squares estimator of $\boldsymbol{\Pi}_v$ takes the form

$$\widehat{\boldsymbol{\Pi}}_v = \left[\boldsymbol{B}_v^\top \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{A}_v}^{-1} \boldsymbol{B}_v\right]^{-1} \boldsymbol{B}_v^\top \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{A}_v}^{-1} \boldsymbol{Z}_v. \tag{29}$$

[8] introduced the following test statistic along with its asymptotic distribution under H_0:

$$M = \left(\boldsymbol{R}\widehat{\boldsymbol{\Pi}}_v\right)^\top \left[\boldsymbol{R}\left(\boldsymbol{B}_v^\top \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{A}_v}^{-1}\boldsymbol{B}_v\right)^{-1}\boldsymbol{R}^\top\right]^{-1}\left(\boldsymbol{R}\widehat{\boldsymbol{\Pi}}_v\right) \overset{A}{\sim} \chi^2_{d^2k}. \tag{30}$$

Note that H_0 is rejected for large values of the test statistic.

The dissimilarity between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ can be assessed by using the p-value associated with the test statistic (30), i.e., by considering

$$d_{VAR,p}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = 1 - P\left(\chi^2_{d^2k} > M\right). \tag{31}$$

It is worth remarking that the distance $d_{VAR,p}$ satisfies the properties of nonnegativity and symmetry. Thus, it can be used as a dissimilarity measure between MTS. In fact, under the alternative hypothesis, $d_{VAR,p}$ properly informs about the degree of discrepancy between two series, in the sense that we expect that $d_{VAR,p}(\boldsymbol{X}_T, \boldsymbol{Z}_T) > d_{VAR,p}(\boldsymbol{X}_T, \boldsymbol{Y}_T)$ if the generating process of $\boldsymbol{X}_T$ is more similar to that of $\boldsymbol{Y}_T$ than to that of $\boldsymbol{Z}_T$. However, when the null hypothesis is true, the use of $d_{VAR,p}$ as a dissimilarity measure presents some disadvantages. Indeed, if $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ have been generated from the same stochastic process, then $d_{VAR,p}(\boldsymbol{X}_T, \boldsymbol{Y}_T) \sim U[0,1]$. Therefore, the distance can easily take large values even though we are in a situation in which low values are expected. This poses a substantial difference between the proposed dissimilarity and other distance metrics described throughout this paper.

It is interesting to note that, if a hierarchical clustering algorithm is executed by considering the $d_{VAR,p}$-based pairwise distance matrix, then a clustering homogeneity criterion is implicitly provided by setting in advance a threshold significance level α (e.g., 0.05 or 0.01). This way, only those pairs of series with associated p-values greater than α (i.e., $d_{VAR,p} < 1-\alpha$) will be located in the same group. This means that only those series whose underlying patterns are not significantly different at level α will be grouped together. In **mlmts**, a hierarchical clustering algorithm taking as input a matrix of p-values is available via the package **TSclust**.

2.4. Dissimilarities based on dimensionality reduction

This section presents some procedures combining a dimensionality reduction technique with the application of a dissimilarity measure in the transformed space. Like the feature-based approaches, these methods aim at circumventing the high computational complexity associated with MTS objects, but, unlike the former, they do so by reducing the dimensionality of the series through SVD. Usually, the success of these procedures depends on keeping enough amount of information so that the underlying patterns in the collection of MTS are preserved whereas the noise is removed. Frequently, the user sets in advance a rate of explained variability that the transformed dataset must retain. Although this kind of algorithms do not explicitly consider the serial dependence structure of the time series, they are generally capable of preserving the temporal information to some extent. Some of the most relevant metrics including a pre-processing step of dimensionality reduction are provided below.

A distance measure based on the Extended Frobenius norm

[9] proposed the so-called Extended Frobenius norm (*Eros*) as a similarity measure for MTS. Basically, *Eros* computes similarity between two series by comparing the eigenvectors of their covariance matrices weighted by the eigenvalues, i.e., evaluating a weighted distance between their principal components. To be more precise, the ith weight represents an aggregated value (e.g., min, max, mean...) of the variances of the ith principal components of all MTS items in the collection. Hence, the weights change when new data are inserted into or removed from the database.

Let $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ be two d-dimensional time series of length T. Let $\boldsymbol{V}_{\boldsymbol{X}_T}$ and $\boldsymbol{V}_{\boldsymbol{Y}_T}$ be the eigenvector matrices obtained by applying SVD to the covariances matrices of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, $\boldsymbol{\Sigma}_{\boldsymbol{X}_T}$ and $\boldsymbol{\Sigma}_{\boldsymbol{Y}_T}$, respectively. In this way, $\boldsymbol{V}_{\boldsymbol{X}_T} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_d^\top\right]$ and $\boldsymbol{V}_{\boldsymbol{Y}_T} = \left[\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_d^\top\right]$, where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i, i = 1, \ldots, d$, are orthonormal vectors of length d. The set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\}$ is associated with the collection of eigenvalues $\left\{\lambda_1^{\boldsymbol{X}_T}, \ldots, \lambda_d^{\boldsymbol{X}_T}\right\}$, with $\lambda_1^{\boldsymbol{X}_T} \geqslant \lambda_2^{\boldsymbol{X}_T} \geqslant \ldots \geqslant \lambda_d^{\boldsymbol{X}_T}$. According to the value of $\lambda_i^{\boldsymbol{X}_T}$, the information contribution of the ith eigenvector, also called ith principal component, can be known, in the sense that a greater value of λ_i indicates a greater information contribution. Analogous remarks can be made for series $\boldsymbol{Y}_T$. The *Eros* similarity between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is defined as

$$Eros(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sum_{i=1}^{d} w_i | < \boldsymbol{x}_i, \boldsymbol{y}_i > | = \sum_{i=1}^{d} w_i |\cos\theta_i|, \tag{32}$$

where $< \boldsymbol{x}_i, \boldsymbol{y}_i >$ is the inner product of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, $\boldsymbol{w} = (w_1, \ldots, w_d)$ is a vector of weights which is based on the eigenvalues of the MTS dataset and properly normalised such that $\sum_{i=1}^{d} w_i = 1$, and θ_i denotes the angle between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. The range of *Eros* is between 0 and 1, with values close to 1 indicating strong similarity between both MTS.

To normalise the weights, either the raw eigenvalues or the normalized eigenvalues can be used. Specifically, assume that we have a collection of n d-dimensional MTS of length T, $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$. Let $\boldsymbol{\lambda}^{(j)} = \left(\lambda_j^1, \ldots, \lambda_j^d\right)$ be the vector containing the eigenvalues associated with MTS j. The weight vector $\boldsymbol{w}$ can be computed from the raw eigenvalues as

$$w_i = \frac{\frac{1}{n}\sum_{k=1}^{n}\lambda_k^{(i)}}{\frac{1}{n}\sum_{j=1}^{d}\sum_{k=1}^{n}\lambda_k^{(j)}}, \tag{33}$$

$i = 1, \ldots, d$. The vector of weights $\boldsymbol{w}$ can be also computed from the normalized eigenvalues only by replacing the raw eigenvalues in (34) by the normalized eigenvalues, i.e., by considering $\overline{\boldsymbol{\lambda}}^{(j)} = \boldsymbol{\lambda}^{(j)} / \sum_{i=1}^{d}\lambda_j^i$.

Based on previous considerations, a dissimilarity measure between $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ is defined as

$$d_{Eros}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = \sqrt{2 - 2Eros(\boldsymbol{X}_T, \boldsymbol{Y}_T)}, \tag{34}$$

where the vector of weights $\boldsymbol{w}$ is computed according to one of the previously indicated ways. The measure d_{Eros} preserves the similarity relation of *Eros*, that is, if $Eros\left(\boldsymbol{X}_T^{(1)}, \boldsymbol{X}_T^{(2)}\right) > Eros\left(\boldsymbol{X}_T^{(1)}, \boldsymbol{X}_T^{(3)}\right)$, then $d_{Eros}\left(\boldsymbol{X}_T^{(1)}, \boldsymbol{X}_T^{(2)}\right) < d_{Eros}\left(\boldsymbol{X}_T^{(1)}, \boldsymbol{X}_T^{(3)}\right)$. However, d_{Eros} does not satisfy the triangle inequality, although it has been successfully applied for similarity search on MTS datasets, outperforming traditional metrics as the Euclidean distance or DTW.

A distance measure based on a PCA similarity factor

[10] introduced a similarity factor based on Principal Component Analysis (PCA) and the angles between the principal component subspaces. The similarity factor is calculated by using the largest principal components (PCs) of each multivariate series, which are weighted according to their explained variance.

Let θ_{ij} be the angle between the ith PC of $\boldsymbol{X}_T$ and the jth PC of $\boldsymbol{Y}_T$ and assume that the first k principal components were retained such that at least 95% of the variance in each MTS is explained. The PCA similarity factor is expressed as

$$S_{PCA}(\boldsymbol{X}_T,\boldsymbol{Y}_T)=\frac{\sum_{i=1}^{k}\sum_{j=1}^{k}\left(\lambda^i_{\boldsymbol{X}_T}\lambda^j_{\boldsymbol{Y}_T}\right)\cos^2\theta_{ij}}{\sum_{i=1}^{k}\lambda^i_{\boldsymbol{X}_T}\lambda^i_{\boldsymbol{Y}_T}}, \tag{35}$$

where $\lambda^i_{\boldsymbol{X}_T}$ and $\lambda^i_{\boldsymbol{Y}_T}$ are the ith eigenvalues of MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, respectively.

A dissimilarity measure inheriting the properties of S_{PCA} can be defined as

$$d_{PCA}(\boldsymbol{X}_T,\boldsymbol{Y}_T)=1-S_{PCA}(\boldsymbol{X}_T,\boldsymbol{Y}_T). \tag{36}$$

The computation of d_{PCA} is implemented in **mlmts** with the help of the package **evolqg** [49].

A dissimilarity based on the two-dimensional singular value decomposition

A novel algorithm for MTS classification using two-dimensional SVD (2dSVD) was constructed by [50]. The approach relies on two steps: (i) row-row and column-column covariance matrices of MTS elements are computed for feature extraction and (ii) one-nearest-neighbour classifier is used to carry out the classification task through a given dissimilarity measure.

Let $\left\{\boldsymbol{X}_T^{(1)},\ldots,\boldsymbol{X}_T^{(n)}\right\}$ be a dataset of MTS, where each series $\boldsymbol{X}_T^{(i)}\in\mathbb{R}^{T\times d}, i=1,\ldots,n$. Define the averaged row-row covariance matrix $\boldsymbol{R}$ and column-column covariance matrix $\boldsymbol{C}$ as follows:

$$\begin{aligned}\boldsymbol{R}&=\tfrac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{X}_T^{(i)}-\overline{\boldsymbol{X}}_T\right)\left(\boldsymbol{X}_T^{(i)}-\overline{\boldsymbol{X}}_T\right)^\top,\\ \boldsymbol{C}&=\tfrac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{X}_T^{(i)}-\overline{\boldsymbol{X}}_T\right)^\top\left(\boldsymbol{X}_T^{(i)}-\overline{\boldsymbol{X}}_T\right),\end{aligned} \tag{37}$$

where $\overline{\boldsymbol{X}}_T=\frac{1}{n}\sum_{i=1}^{n}\overline{\boldsymbol{X}}_T^{(i)}$. Let $\boldsymbol{U}_r$ be a matrix containing the r principal eigenvectors of $\boldsymbol{R}$ in terms of information contribution as columns, and $\boldsymbol{V}_s$ be a matrix containing the s principal eigenvectors of $\boldsymbol{C}$ as columns, i.e., $\boldsymbol{U}_r=[u_1,\ldots,u_r]\in\mathbb{R}^{T\times r}$ and $\boldsymbol{V}_s=[v_1,\ldots,v_s]\in\mathbb{R}^{d\times s}$. The feature matrix associated with MTS i is expressed as

$$\boldsymbol{M}^{(i)}=\boldsymbol{U}_r^\top\boldsymbol{X}_T^{(i)}\boldsymbol{V}_s, \tag{38}$$

$i=1,\ldots,n$, with $\boldsymbol{M}^{(i)}\in\mathbb{R}^{r\times s}$.

Denoting by $\boldsymbol{M}^{(i)}_{\bullet j}$ the jth column of matrix $\boldsymbol{M}^{(i)}$, the distance between two series in the collection, $\boldsymbol{X}_T^{(j)}$ and $\boldsymbol{X}_T^{(k)}$, is defined as

$$d_{2dSVD}\left(\boldsymbol{X}_T^{(j)},\boldsymbol{X}_T^{(k)}\right)=\sum_{b=1}^{s}\left\|\boldsymbol{M}^{(j)}_{\bullet b}-\boldsymbol{M}^{(k)}_{\bullet b}\right\|. \tag{39}$$

It is worth highlighting that [50] show how the distance d_{2dSVD} outperforms a dissimilarity constructed from one-dimensional SVD (1d-SVD) in the context of supervised classification.

A metric based on locality preserving projections

[20] introduced a new method for MTS classification based on locality preserving projections (LPP), which is a linear projection map that optimally preserves the neighbourhood structure of the dataset. By performing LPP, the MTS samples are projected into a lower dimensional space in which the series related to the same class are close to each other. The approach is based on three steps: (i) each series is replaced by a feature vector, (ii) the feature vectors are projected into a lower dimensional space via LPP and (iii) a distance measure is computed in the transformed space to carry out the classification task.

Let $\left\{\boldsymbol{X}_T^{(1)},\ldots,\boldsymbol{X}_T^{(n)}\right\}$ be a dataset of MTS, where each series $\boldsymbol{X}_T^{(i)}\in\mathbb{R}^{T\times d}, i=1,j=1,\ldots,n$ and let $\boldsymbol{\Sigma}=\left\{\boldsymbol{\Sigma}_{\boldsymbol{X}_T^{(1)}},\ldots,\boldsymbol{\Sigma}_{\boldsymbol{X}_T^{(n)}}\right\}$ be the corresponding set of covariance matrices. Transforming the set $\boldsymbol{\Sigma}$ through SVD, each MTS can be associated with a matrix of d orthonormal singular vectors, $\boldsymbol{V}^{(i)}=\left[\boldsymbol{v}_{i,1}^\top,\ldots,\boldsymbol{v}_{i,d}^\top\right]$, and a vector of normalized singular values, $\tilde{\boldsymbol{\lambda}}^{(i)}=\frac{\boldsymbol{\lambda}^{(i)}}{||\boldsymbol{\lambda}^{(i)}||}=\left(\tilde{\lambda}_1^{(i)},\ldots,\tilde{\lambda}_d^{(i)}\right)$, with $i=1,\ldots,n$. Based on previous remarks, each series can by represented by means of a vector in one of the following ways:

1. Replacing each $\boldsymbol{X}_T^{(i)}$ in the original collection by the vector $\boldsymbol{\varphi}^{(i)}=\left(\boldsymbol{v}_{i,1},\tilde{\boldsymbol{\lambda}}_1^{(i)},\ldots,\tilde{\boldsymbol{\lambda}}_d^{(i)}\right)$ [18].
2. Replacing each $\boldsymbol{X}_T^{(i)}$ in the original collection by the vector $\boldsymbol{\varphi}^{(i)}=(w_{i,1}\boldsymbol{v}_{i,1},w_{i,2}\boldsymbol{v}_{i,2})$, where $w_{i,1}=\frac{\tilde{\lambda}_i^1}{\sum_{j=1}^{d}\tilde{\lambda}_i^j}$ and $w_{i,2}=\frac{\tilde{\lambda}_i^2}{\sum_{j=1}^{d}\tilde{\lambda}_i^j}$ [19]. This way, similarity relies on the closeness between the two more informative directions, properly weighted by their normalised variances.

Once the feature extraction step is performed, the original collection of series is replaced by the set of vectors $\{\boldsymbol{\varphi}^{(i)},\ldots,\boldsymbol{\varphi}^{(n)}\}$, which can be arranged as columns to form the matrix $\boldsymbol{\varphi}=\left[\boldsymbol{\varphi}^{(i)\top},\ldots,\boldsymbol{\varphi}^{(n)\top}\right]$.

The next step involves the execution of LPP, whose objective funtion is given by

$$\min_{\boldsymbol{a}}\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{\varphi}^{(i)}\boldsymbol{a}^\top-\boldsymbol{\varphi}^{(j)}\boldsymbol{a}^\top)S_{ij}, \tag{40}$$

where $\boldsymbol{a}$ is the transformation vector and $\boldsymbol{S}=(S_{ij})$ is a matrix assessing the local structure of the set of feature vectors. A possible way of defining $\boldsymbol{S}$ is as follows: $S_{ij}=\exp\frac{\|\boldsymbol{\varphi}^{(i)}-\boldsymbol{\varphi}^{(j)}\|}{b}$ if $\boldsymbol{\varphi}^{(i)}$ is among the k nearest neighbours of $\boldsymbol{\varphi}^{(j)}$ or $\boldsymbol{\varphi}^{(j)}$ is among the k nearest neighbours of $\boldsymbol{\varphi}^{(i)}$, where b is a suitable constant. Otherwise, $S_{ij}=0$. The transformation vector $\boldsymbol{a}$ that minimizes the objective function (40) is determined by the minimum eigenvalue solution to the generalized eigenvalue problem

$$\boldsymbol{\varphi}\boldsymbol{L}\boldsymbol{\varphi}^\top\boldsymbol{a}^\top=\lambda\boldsymbol{\varphi}\boldsymbol{D}\boldsymbol{\varphi}^\top\boldsymbol{a}^\top, \tag{41}$$

where $\boldsymbol{D}=(D_{ij})$ is a diagonal matrix such that $D_{ii}=\sum_{j=1}^{n}S_{ji}$, and $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{S}$ is the Laplacian matrix.

Let vectors $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_d$ be the solution of (41) and define $\boldsymbol{A}_{LPP}=[\boldsymbol{a}_1^\top,\ldots,\boldsymbol{a}_d^\top]$. A distance measure between two series in the original collection, $\boldsymbol{X}_T^{(j)}$ and $\boldsymbol{X}_T^{(k)}$, can be defined as

$$d_{LPP}\left(\boldsymbol{X}_T^{(j)},\boldsymbol{X}_T^{(k)}\right)=\left\|\boldsymbol{\varphi}^{(j)}\boldsymbol{A}_{LPP}-\boldsymbol{\varphi}^{(k)}\boldsymbol{A}_{LPP}\right\|. \tag{42}$$

[20] show that the distance measure d_{LPP} performs comparably to d_{2dSVD} in terms of classification error rate and CPU runtime. The advantage of the former distance is that it can be used with MTS samples of different lengths, whereas d_{2dSVD} assumes that all MTS items have the same length. The main challenge related to the distance d_{LPP} lies in the selection of the dimensionality of LPP subspace and the number of neighbours for the computation of matrix $\boldsymbol{S}$.

A distance constructed via a piecewise representation based on PCA

[51] proposed to reduce the dimensionality of a given MTS by considering an extension of standard PCA which takes into account the local information contained in the series. The method, called piecewise representation based on PCA (PPCA), segments a MTS into several sequences and employs PCA to obtain the principal components with respect to an average covariance matrix considering independently each one of the sequences. Package **mlmts** introduces a dissimilarity measure based on PPCA. Starting from an MTS $\boldsymbol{X}_T\in\mathbb{R}^{T\times d}$, assumed to be centered (its columns add up to zero), the technique PPCA is based on the following steps:

1. Split the series $\boldsymbol{X}_T \in \mathbb{R}^{T\times d}$ into w segments of approximately the same length along the time direction, $\boldsymbol{X}_{T,1},\ldots,\boldsymbol{X}_{T,w}$. Note that each $\boldsymbol{X}_{T,i}, i=1,\ldots,w$ is a MTS with d variables and approximate length T/w.
2. For each segment $\boldsymbol{X}_{T,i}$, compute the corresponding covariance matrix $\boldsymbol{\Sigma}_i, i=1,\ldots,w$. Construct the average covariance matrix $\boldsymbol{\Sigma}_a = \frac{1}{w}\sum_{i=1}^{w}\boldsymbol{\Sigma}_i$.
3. Use SVD to decompose matrix $\boldsymbol{\Sigma}_a$ as $\boldsymbol{V}_a\boldsymbol{\Lambda}_a\boldsymbol{V}_a^\top$, where $\boldsymbol{\Lambda}_a$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{\Sigma}_a$ and $\boldsymbol{V}_a$ is a orthonormal matrix containing the eigenvectors of $\boldsymbol{\Sigma}_a$ as columns. Let $\boldsymbol{V}_a(r)\in\mathbb{R}^{d\times r}$ be a matrix containing the r principal eigenvectors of $\boldsymbol{\Sigma}_a$ in terms of information contribution as columns.
4. Compute the reduced series $\boldsymbol{M}_{\boldsymbol{X}_T} = \boldsymbol{X}_T\boldsymbol{V}_a(r)\in\mathbb{R}^{T\times r}$.

Note that the previous method encompasses a dimensionality reduction procedure employing the standard PCA when $w=1$. Advantages of the local approach with respect to the global one include a more detailed representation of the series, a better ability to handle large MTS (where relationships between variables usually become complex over time) and a greater effectiveness when performing dimensionality reduction in real MTS datasets.

For two series $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, a distance measure based on PPCA is defined as

$$d_{PPCA}(\boldsymbol{X}_T,\boldsymbol{Y}_T) = ||vec(\boldsymbol{\Sigma}_a^{\boldsymbol{X}_T}) - vec(\boldsymbol{\Sigma}_a^{\boldsymbol{Y}_T})||, \tag{43}$$

where the superscripts $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$ indicate the series used to construct the average covariance matrix. It is assumed that $\boldsymbol{\Sigma}_a^{\boldsymbol{X}_t}$ and $\boldsymbol{\Sigma}_a^{\boldsymbol{Y}_t}$ are computed by considering analogous segments in both series (i.e., $\boldsymbol{X}_{T,i}$ has the same length than $\boldsymbol{Y}_{T,i}$ for all $i=1,\ldots,w$). By definition, d_{PPCA} assesses discrepancy between representations of the local structures of $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$. Therefore, this distance could be particularly useful with datasets of long time series exhibiting complicated relations over time.

A spatial dissimilarity relying on Variable-Based Principal Component Analysis

[52] devised a novel dissimilarity measure between multivariate series which preserves the 2-dimensional structure of MTS samples, thus avoiding their vectorization. Its computation consists of two stages. First, a new dimensionality reduction technique based on variable-based Principal Component Analysis (VPCA) is performed to reduce the dimensionality of MTS items in the original collection. Afterwards, a spatial weighted matrix distance (SWMD) is considered to assess discrepancy between pairs of transformed series. The SWMD evaluates proximity between raw values of two reduced MTS but taking into account the spatial correlations between different coordinates in the 2-D space.

Let $\left\{\boldsymbol{X}_T^{(1)},\ldots,\boldsymbol{X}_T^{(n)}\right\}$ a dataset of n d-dimensional MTS of length T, where $\boldsymbol{X}_T^{(i)} = \left[\boldsymbol{X}_{T,1}^{(i)\top},\ldots,\boldsymbol{X}_{T,d}^{(i)\top}\right]$ is a $T\times d$ matrix whose columns are the univariate series $\boldsymbol{X}_{T,j}^{(i)}, j=1,\ldots,d$. To reduce the dimensionality of MTS items through VPCA, first we combine the records of every variable from all the samples into one matrix, i.e., we define

$$\boldsymbol{V}_j = \begin{bmatrix} \boldsymbol{X}_{T,j}^{(1)} \\ \boldsymbol{X}_{T,j}^{(2)} \\ \ldots \\ \boldsymbol{X}_{T,j}^{(n)} \end{bmatrix}, \ j=1,\ldots,d. \tag{44}$$

Note that $\boldsymbol{V}_j\in\mathbb{R}^{n\times T}$ is a matrix whose kth row contains the jth UTS within the kth MTS. In this way, matrix $\boldsymbol{V}_j$ is associated with the jth variable in the original collection. Dimensionality reduction is carried out as follows. For each $1\leqslant j\leqslant d$, the eigenvectors and eigenvalues of the covariance matrix of $\boldsymbol{V}_j$, denoted by $\boldsymbol{\Sigma}_{\boldsymbol{V}_j}$, are obtained and the smallest number p_j of eigenvalues ensuring a given cumulative rate of explained variability Q is selected. Then, we set $p^* = \min_{1\leqslant j\leqslant d}\{p_j\}$ and, for each j, construct the matrix $\boldsymbol{U}_{\boldsymbol{V}_j} = \left[\boldsymbol{u}_{j,1},\ldots,\boldsymbol{u}_{j,p^*}\right]$ formed by the p^* first eigenvectors of $\boldsymbol{\Sigma}_{\boldsymbol{V}_j}$. Projecting each $\boldsymbol{V}_j$ by $\boldsymbol{U}_{\boldsymbol{V}_j}$ to a space with lower dimensionality, we obtain

$$\boldsymbol{F}_j = \boldsymbol{V}_j\boldsymbol{U}_{\boldsymbol{V}_j} = \begin{bmatrix} \boldsymbol{f}_{j,1} \\ \boldsymbol{f}_{j,2} \\ \ldots \\ \boldsymbol{f}_{j,n} \end{bmatrix}, \ j=1,\ldots,d, \tag{45}$$

where $\boldsymbol{f}_{j,k}$ is the kth row of $\boldsymbol{F}_j$. Note that $\boldsymbol{F}_j\in\mathbb{R}^{n\times p^*}$. The ith sample MTS, $\boldsymbol{X}_T^{(i)}$, is then reconstructed from the set $\{F_j, j=1,\ldots,d\}$ to obtain a dimensionality-reduced MTS $\boldsymbol{Y}_i$ proceeding as follows:

$$\boldsymbol{Y}_i = \left[\boldsymbol{f}_{1,i}^\top,\boldsymbol{f}_{2,i}^\top,\ldots,\boldsymbol{f}_{d,i}^\top\right], \ i=1,\ldots,n, \tag{46}$$

with $\boldsymbol{Y}_i\in\mathbb{R}^{p^*\times d}$. It is worth remarking that the kth column of $\boldsymbol{Y}_i$ is obtained according to the variation distribution of all kth dimensions in the original collection of multivariate series. By replacing the initial dataset $\left\{\boldsymbol{X}_T^{(i)}\right\}_{i=1}^n$ by $\{\boldsymbol{Y}_j\}_{j=1}^n$, the length of the original series is reduced from T to p^*.

The SWMD between two series $\boldsymbol{X}_T^{(j)}$ and $\boldsymbol{X}_T^{(k)}$ is defined in terms of their reduced counterparts $\boldsymbol{Y}_j$ and $\boldsymbol{Y}_k$ in the following way:

$$d_{SWMD}\left(\boldsymbol{X}_T^{(j)},\boldsymbol{X}_T^{(k)}\right) = \left[\left(vec(\boldsymbol{Y}_j) - vec(\boldsymbol{Y}_k)\right)\boldsymbol{S}\left(vec(\boldsymbol{Y}_j) - vec(\boldsymbol{Y}_k)\right)^\top\right]^{1/2}, \tag{47}$$

where $\boldsymbol{S} = (S_{uw})$ is a spatial matrix which captures the intrinsic correlations of the coordinates in $\mathbb{R}^{p^*\times d}$. In fact, S_{uw} represents the spatial dimensionality distance between the uth element of $vec(\boldsymbol{Y}_j)$ and the wth element of $vec(\boldsymbol{Y}_k)$.

Let $\boldsymbol{y}_k^i$ be the ith element of $vec(\boldsymbol{Y}_k), i=1,\ldots,p^*d, k=1,\ldots,n$. Note that $\boldsymbol{y}_k^i$ is associated with a given row and a given column of $\boldsymbol{Y}_k$, denoted by $r(i)$ and $c(i)$, respectively. In this way, the dimensionality distance between $\boldsymbol{y}_j^u$ and $\boldsymbol{y}_k^w$ is expressed as

$$d_D\left(\boldsymbol{y}_j^u,\boldsymbol{y}_k^w\right) = d_D(u,w) = \left[(r(u)-r(w))^2 + (c(u)-c(w))^2\right]^{1/2}. \tag{48}$$

The distance $d_D\in\left[0,\sqrt{(d-1)^2+(p^*-1)^2}\right]$ and takes small values when $\boldsymbol{y}_j^u$ and $\boldsymbol{y}_k^w$ are located in close rows/columns. Otherwise, the distance d_D takes large values. The matrix $\boldsymbol{S}$ can be defined as a monotonically decreasing function of d_D as follows:

$$S_{uw} = \frac{1}{2\pi(1-p^{*-1})^2}\exp\left(-\frac{d_D^2(u,w)}{2(1-p^{*-1})^2}\right). \tag{49}$$

Below we give two remarks concerning the construction of the matrix $\boldsymbol{S}$:

- According to (47), S_{uw} can be seen as the weight associated with the factor $\left(\boldsymbol{y}_j^u-\boldsymbol{y}_k^u\right)\left(\boldsymbol{y}_j^w-\boldsymbol{y}_k^w\right)$ in the computation of the distance d_{SWMD}. This weight gets larger as the distance $d_D(u,w)$ gets smaller, and decreases as the position difference of uth and wth elements increases. Therefore, the dissimilarity d_{SWMD} can be regarded as a weighted Euclidean distance, which directly includes the spatial dimensionality difference between the corresponding elements.

- The variation range of d_D shrinks as p^* decreases. Therefore, for very small values of p^*, the range of d_D is substantially narrow. The quantity $(1-p^{*-1})^2$ is incorporated in (49) to avoid that S_{uw} inherits the restricted range of d_D. Indeed, this factor provokes that S_{uw} keeps a sufficiently wide range so that the spatial distance between the uth and wth elements is correctly identified in the feature matrix space even for small values of p^*.

According to (47) and (49), the distance d_{SWMD} can be also expressed as

$$d^2_{SWMD}\left(\boldsymbol{X}_T^{(j)}, \boldsymbol{X}_T^{(k)}\right) = \frac{1}{2\pi(1-p^{*-1})^2} \sum_{u,w=1}^{p^*d} \exp\left(-\frac{d_D^2(u,w)}{2(1-p^{*-1})^2}\left(\boldsymbol{y}_j^u - \boldsymbol{y}_k^u\right)\left(\boldsymbol{y}_j^w - \boldsymbol{y}_k^w\right)\right). \quad (50)$$

The advantages of d_{SWMD} in the context of soft clustering of MTS are shown in [52]. Specifically, it is argued that VPCA allows to extract more information from the MTS collection than other classical dimensionality reduction techniques. Furthermore, by considering the spatial weighted matrix $\boldsymbol{S}$, the inherent structural information of MTS samples is preserved, which is not possible with traditional dissimilarities as the Euclidean distance.

A dissimilarity based on maximal cross-correlations

A dimensionality reduction technique employing maximal cross-correlations was introduced by [53]. The procedure is characterized by two key points: (i) a novel approach to measure cross-correlation between each pair of components of a MTS, and (ii) a graph-based clustering algorithm using the quantities obtained in (i) as input. Besides implementing the method, package **mlmts** takes advantage of this procedure to define a new dissimilarity between MTS. In this way, the use of the corresponding function is not limited to lowering the dimension of a given series. Given the MTS $\boldsymbol{X}_T \in \mathbb{R}^{T\times d}$, [53] propose to remove the redundant components by means of the following steps:

1. For each pair of components (j_1, j_2) in $\boldsymbol{X}_T$, compute the set $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{j_1j_2} = \{|\widehat{\rho}^{\boldsymbol{X}_T}_{j_1j_2}(-L)|, \ldots, |\widehat{\rho}^{\boldsymbol{X}_T}_{j_1j_2}(-1)|, |\widehat{\rho}_{j_1, j_2^{\boldsymbol{X}_T}}(0)|, |\widehat{\rho}^{\boldsymbol{X}_T}_{j_1j_2}(1)|, \ldots, |\widehat{\rho}^{\boldsymbol{X}_T}_{j_1j_2}(L)|\}$, with L being the maximum number of lags fixed beforehand by the user. Define $M\left(\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{j_1j_2}\right) = \max\left(\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{j_1j_2}\right)$.
2. Construct a symmetric matrix $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}$ whose entries are the quantities $M\left(\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{j_1j_2}\right), j_1, j_2 = 1, \ldots, d$. Note that the diagonal elements of $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}$ are all equal to one. This matrix can be seen as the adjacency matrix of a weighted, undirected graph.
3. Given a threshold $\delta \in (0,1)$, construct a new matrix $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{\delta}$ by keeping the entries of $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}$ above δ and setting all the remaining entries to zero.
4. Compute the connected components of the graph defined by matrix $\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}_{\delta}$, which can be seen as groups of related variables. For each component, define the corresponding center as the element maximizing the sum of the weights of its edges. Note that each center corresponds to a specific variable in MTS $\boldsymbol{X}_T$.
5. Construct the reduced MTS, $\boldsymbol{X}_T^*$, by selecting the variables of $\boldsymbol{X}_T$ corresponding to the center elements obtained in Step 4.

The previous procedure can be applied independently to each MTS in a given dataset, producing a collection of MTS with lower dimensionality but not necessarily having the same number of variables. Each reduced MTS is expected to contain fewer components than its original counterpart while retaining the most important patterns exhibited by the latter. Hence, by considering the transformed dataset instead of the original one, the user could dramatically improve the computation times of several machine learning algorithms.

Note that a distance between MTS usually requires both series to have the same number of dimensions, so a standard metric cannot be computed in the collection of reduced series. Based on previous considerations, given two MTS $\boldsymbol{X}_T$ and $\boldsymbol{Y}_T$, a dissimilarity measure based on maximal cross-correlations can be defined as

$$d_{MCC}(\boldsymbol{X}_T, \boldsymbol{Y}_T) = ||vec\left(\widehat{\boldsymbol{\Theta}}^{\boldsymbol{X}_T}\right) - vec\left(\widehat{\boldsymbol{\Theta}}^{\boldsymbol{Y}_T}\right)||. \quad (51)$$

It is worth remarking that d_{MCC} and the correlation-based distance d_{COR} in (13) differ in two ways: (i) unlike d_{COR}, d_{MCC} does not measure dissimilarity between estimated autocorrelations, and (ii) d_{MCC} only assesses discrepancy between the maximum values of the cross-correlations for each pair (j_1, j_2). Therefore, two MTS could be considered similar with d_{MCC} even if their serial cross-dependence patterns occur at very different lags.

2.5. Specific machine learning tools

The metrics introduced in Section 2 and included in **mlmts** can be used to perform several machine learning techniques on a given set of n MTS, $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$. For instance, a pairwise dissimilarity matrix constructed from the collection of series can be used to execute a clustering algorithm. The same distance matrix could be also used to detect outlier series, i.e., series displaying the largest average distance to the remaining series in the dataset. In addition, a classification procedure based on kNN could be carried out by computing the distances between a given element in the test set and every element in the corresponding training set. In sum, many powerful function available through R packages can be used with output produced by **mlmts**. Nevertheless, some particular tools for MTS data mining have been implemented in **mlmts**. These procedures can not be directly executed from any of the dissimilarities presented in Section 2, so their availability to the package broadens the variety of methods made accessible to the user. An overview of these specific methods is provided in this section.

A classification algorithm based on QCD and MODWT

[17] proposed the so-called Fast Forest of Flexible Features (F4) algorithm for MTS classification. This classifier consists of two stages. First, an array of features based on QCD and MODWT are extracted from each series, and the classical PCA is performed with respect to the set of features associated with QCD. Second, a traditional random forest is fed with the extracted features.

Assume we have a collection of n MTS $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$ such that the first n_1 are labelled, whereas the remaining series are unlabelled. The classification task consists of learning an algorithm from the set of labelled series so that the class labels associated with the set of unlabelled MTS can be predicted with high accuracy. Usually, in the context of supervised classification, the collection $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n_1)}\right\}$ is called the training set, whereas the collection $\left\{\boldsymbol{X}_T^{(n_1+1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$ is referred to as the test set. An outline of the steps executed by the classifier F4 is provided below (see Section 2 in [17] for further details).

1. A vector of QCD-based features is extracted from each element in the set $\left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$. Specifically, given the series $\boldsymbol{X}_T^{(i)}, i = 1, \ldots, n$ in the original set, the vectorized version of the array $\left\{W\left[\widehat{G}_{j_1}, \; j_2^{\boldsymbol{X}_T^{(i)}}(\omega, \tau, \tau')\right] : W \in \{\Re, \Im\}, \; 1 \leqslant j_1, j_2 \leqslant d, \; \omega \in \Omega, \; \tau, \tau' \in \mathscr{T}\right\}$ is considered. We denote this vector by

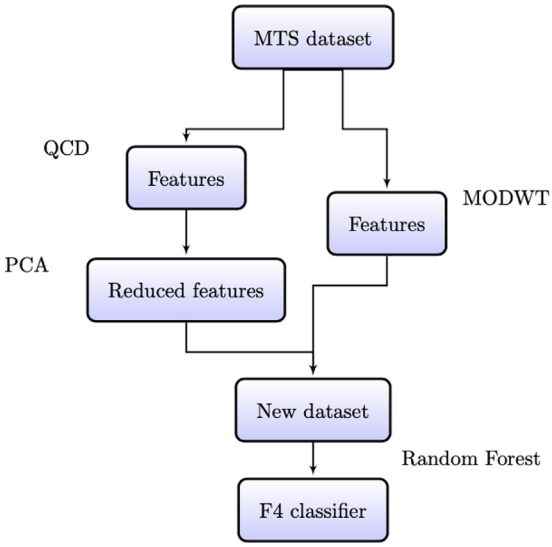


Fig. 1. Flowchart of F4 classifier.

$\mathbf{\Psi}^{(i)}_{QCD}$. In this way, each series is described by means of real and imaginary parts of its smoothed CCR-periodograms evaluated in every pair of components, every pair of quantile levels from a pre-fixed set, and the set of Fourier frequencies. This feature extraction procedure results in the set $\mathbf{\Psi}_{QCD} = \left\{\mathbf{\Psi}^{(1)}_{QCD}, \ldots, \mathbf{\Psi}^{(n)}_{QCD}\right\}$.

2. A vector of MODWT-based features is extracted from each element in the set $\left\{\boldsymbol{X}^{(1)}_T, \ldots, \boldsymbol{X}^{(n)}_T\right\}$. Specifically, given the series $\boldsymbol{X}^{(i)}_T, i = 1, \ldots, n$ in the original collection, the vectorized version of the set $WV \cup WC$ is considered, where $WV = \left\{\widehat{\nu}^2_{\boldsymbol{X}^{(i)}_{Tj}}(\nu_k) : j = 1, \ldots, d,\ k = 1, \ldots, J\right\}$ and $WC = \left\{\widehat{\psi}_{\boldsymbol{X}^{(i)}_{Tj_1}\boldsymbol{X}^{(i)}_{Tj_2}}(\nu_k) : j_1, j_2 = 1, \ldots, d,\ j_1 > j_2,\ k = 1, \ldots, J\right\}$. In this way, each series is described by means of its associated wavelet variances and wavelet correlations. This feature extraction procedure results in the set $\mathbf{\Psi}_{MODWT} = \left\{\mathbf{\Psi}^{(1)}_{MODWT}, \ldots, \mathbf{\Psi}^{(n)}_{MODWT}\right\}$.
3. The classical PCA is applied to the dataset associated with the collection $\mathbf{\Psi}_{QCD}$. After retaining the p first principal components, the set of score vectors $\mathbf{\Psi}_{QCD,PCA} = \left\{\mathbf{\Psi}^{(1)}_{QCD,PCA}, \ldots, \mathbf{\Psi}^{(n)}_{QCD,PCA}\right\}$ is obtained. Note that each vector of the form $\mathbf{\Psi}^{(i)}_{QCD,PCA}$ is a vector of length p.
4. Each series in the original collection, $\boldsymbol{X}^{(i)}_T$, is described by means of the vector $\mathbf{\Psi}^{(i)}_{QCD,MODWT} = \left(\mathbf{\Psi}^{(i)}_{QCD,PCA}, \mathbf{\Psi}^{(i)}_{MODWT}\right), i = 1, \ldots, n$.
5. A traditional random forest is trained in the dataset associated with the collection of vectors $\left\{\mathbf{\Psi}^{(1)}_{QCD,MODWT}, \ldots, \mathbf{\Psi}^{(n_1)}_{QCD,MODWT}\right\}$.
6. For each vector $\mathbf{\Psi}^{(i)}_{QCD,MODWT}, i = n_1 + 1, \ldots, n$, the corresponding predicted class label is computed by using the random forest constructed in the previous step.

Fig. 1 shows a flowchart of F4 classifier.

Now we provide some remarks concerning classifier F4. First, several hyperparameters have to be set in advance before executing the procedure. Some of the most important hyperparameters are the pre-fixed set of probability levels, $\mathscr{T}$, and the number of selected principal components, p. The numerical experiments carried out in [17] showed that a small numbers of probability levels regularly spaced on $[0, 1]$ (e.g., $\mathscr{T} = \{0.1, 0.5, 0.9\}$) and a number of principal components such that at least 90% of the variability of the dataset is retained are enough to reach satisfactory results. The default options of these input parameters provided in the **mlmts** version of F4 are based on previous considerations. Second, note that the feature extraction phase, including the application of the PCA transformation, involves all the series and not only the training ones. This is not an issue, since no information about the class labels is necessary to carry out the corresponding stage. In fact, employing the whole set of MTS for feature extraction is necessary, since the test elements must be defined in the same space as the training elements in order to compute proper predictions (step 6 before). Lastly, it is worth highlighting that classifier F4 was thoroughly analysed in [17] by using synthetic and real data and two important conclusions were obtained: (i) F4 obtains, on average, better results than other state-of-the-art algorithms and (ii) the consideration of the PCA transformation constitutes a key step in the success of F4.

An outlier detection procedure based on QCD

[26] introduced a novel method for detecting outlier samples in a multivariate time series dataset, so-called Quantile Cross-Spectral Density Functional for Outlier Detection (QCD-F-OD). The approach first describes each MTS by means of its corresponding smoothed CCR-periodograms, which are treated as a multivariate functional datum. Then an outlier score is assigned to each MTS by using functional depths. In this way, the most anomalous series in the collection are those associated with the lowest depth values. Starting from the set $\left\{\boldsymbol{X}^{(1)}_T, \ldots, \boldsymbol{X}^{(n)}_T\right\}$, the outlier detection procedure proposed in [26] relies on the following steps (see Section 2 in [26] for further details):

1. The collection of multivariate series $\left\{\boldsymbol{X}^{(1)}_T, \ldots, \boldsymbol{X}^{(n)}_T\right\}$ is replaced by the collection of multivariate functional data $\left\{\mathscr{X}^{(1)}(\omega), \ldots, \mathscr{X}^{(n)}(\omega)\right\}$, where each $\mathscr{X}^{(i)}(\omega), i = 1, \ldots, n$, is formed by several univariate functions. Particularly, given the series $\boldsymbol{X}^{(i)}_T$, the element $\mathscr{X}^{(i)}(\omega)$ is formed by the set of curves $\left\{W\left[\widehat{G}^{\boldsymbol{X}^{(i)}_T}_{j_1 j_2}(\omega, \tau, \tau')\right] : W \in \{\Re, \Im\},\ 1 \leqslant j_1, j_2 \leqslant d, \tau, \tau' \in \mathscr{T} = \{\tau_1, \ldots, \tau_r\}\right\}$, where $\omega \in [0, \pi]$. Note that, in this way, each MTS is represented by means of a set of $2d^2r^2$ curves which are function of the frequency ω.
2. For each element $\mathscr{X}^{(i)}(\omega), i = 1, \ldots, n$, its multivariate functional depth with respect to the set $\left\{\mathscr{X}^{(1)}(\omega), \ldots, \mathscr{X}^{(n)}(\omega)\right\}$, denoted by $D_M\left(\mathscr{X}^{(i)}(\omega)\right)$, is computed as a sum of univariate depths. That is, if $\mathscr{X}^{(i)}(\omega) = \left\{\mathscr{X}^{(1)}_i(\omega), \ldots, \mathscr{X}^{(2d^2r^2)}_i(\omega)\right\}$, where the $\mathscr{X}^{(j)}(\omega)$ are univariate functions of ω, then $D_M\left(\mathscr{X}^{(i)}(\omega)\right) = D^1_i + \ldots + D^{2d^2r^2}_i$, where D^j_i is the univariate functional depth of curve $\mathscr{X}^{(j)}_i(\omega)$ with respect to the set $\left\{\mathscr{X}^{(j)}_1(\omega), \ldots, \mathscr{X}^{(j)}_n(\omega)\right\}$.
3. Given the set of depths $\left\{D_M\left(\mathscr{X}^{(1)}(\omega)\right), \ldots, D_M\left(\mathscr{X}^{(n)}(\omega)\right)\right\}$, its elements are sorted in increasing order, obtaining the vector V_{D_M}. For a given desired proportion of outliers to detect, $\alpha \in [0, 1]$, the MTS associated with the first $\lceil \alpha n \rceil$elements, denoting $\lceil \cdot \rceil$ the ceiling function, are determined to be anomalous series.

Fig. 2 shows a flowchart of the outlier detection algorithm.

In what follows we give some comments on the outlier detection procedure. First, the technique requires the selection of some hyperparameters, namely the set of probability levels $\mathscr{T}$ (Step 1 before), the univariate functional depth D (Step 2 before) and the desired proportion of outliers α (Step 3 before). Concerning the former two parameters, the numerical analyses performed in [26]

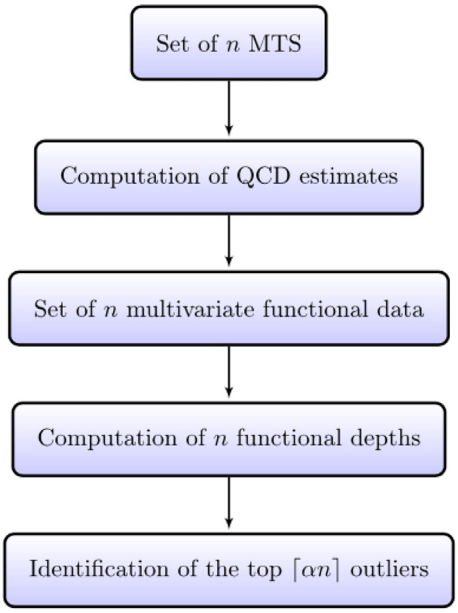


Fig. 2. Flowchart of the outlier detection procedure.

revealed that $\mathscr{T} = \{0.1, 0.5, 0.9\}$ is enough to attain adequate results, and that the Fraiman-Muniz depth [54] outperforms alternative functional depths in terms of outlier detection accuracy. With respect to the rate α, it can be set in advance as long as the user knows an approximate desired number of outliers to detect. Alternatively, one can analyse the distribution of the computed depths to determine the optimal proportion of outliers. Second, the time complexity of the algorithm can be approximated by $O\left(nTd^2\right)$, thus behaving linearly with respect to the number of MTS and the series length and quadratically with respect to the number of dimensions. Finally, it is worth highlighting that outlier identification technique was extensively examined in [26] by means of a broad simulation study and two significant conclusions were reached: (i) the procedure outperforms alternative anomaly detection algorithms proposed in the literature and (ii) the treatment of the smoothed CCR-periodograms as functional data is substantially advantageous for outlier identification.

A clustering algorithm based on Common Principal Component Analysis

A fast and accurate algorithm for crisp clustering of MTS based on Common Principal Component Analysis (CPCA) was introduced by [55]. The procedure, so-called Mc2PCA, is inspired by the traditional *K*-means clustering method and constructs a common projection axes as prototype of each cluster. In this way, the reconstruction error of each MTS projected on the corresponding common projection axes is used to reassign the members of the cluster. A nice property of Mc2PCA is that it considers the relationship between dimensions and the distribution of the entries in the original MTS. Below we give a description of Mc2PCA algorithm (for more details, see [55]).

Suppose we have a dataset having n MTS of length T and dimension d, $\boldsymbol{X} = \left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$, where each $\boldsymbol{X}_T^{(i)} \in \mathbb{R}^{T\times d}$. To obtain the common principal components of dataset $\boldsymbol{X}$, we consider the collection of covariance matrices $\left\{\boldsymbol{\Sigma}_1^*, \ldots, \boldsymbol{\Sigma}_n^*\right\}$, where each $\boldsymbol{\Sigma}_i^* = E\left[\boldsymbol{X}_{\boldsymbol{T}}^{*(i)\top}\boldsymbol{X}_T^{*(i)}\right]$, being $\boldsymbol{X}_{\boldsymbol{T}}^{*(i)}$ the normalized matrix of $\boldsymbol{X}_{\boldsymbol{T}}^{(i)}, i = 1, \ldots, n$, in which the mean of the corresponding column has been subtracted from each entry. Next, all the covariance matrices can be averaged to a common covariance matrix $\overline{\boldsymbol{\Sigma}^*}$ according to

$$\overline{\boldsymbol{\Sigma}^*} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Sigma}_i^*. \tag{52}$$

After decomposing the common covariance matrix $\overline{\boldsymbol{\Sigma}^*}$ through SVD, the first p eigenvectors $u_1, \ldots, u_p$ can be selected to constructed the matrix $\boldsymbol{U}_p = [u_1, \ldots, u_p]$, with $p \leqslant d$. $\boldsymbol{U}_p$ is the common space constructed through CPCA by considering the whole MTS dataset. In fact, the space $\boldsymbol{U}_p$, usually referred to as common projection axes, is expected to describe the features of the MTS better than the original space, since $\boldsymbol{U}_p$ has lower dimension than the latter but retaining most of the data information. Each MTS $\boldsymbol{X}_T^{(i)}$ can then be projected into $\boldsymbol{U}_p$ to obtain

$$\boldsymbol{P}_i = \boldsymbol{X}_T^{(i)}\boldsymbol{U}_p, \ i = 1, \ldots, n, \tag{53}$$

with $\boldsymbol{P}_i \in \mathbb{R}^{T\times p}$. Note that $\boldsymbol{P}_i$ is the representation of $\boldsymbol{X}_T^{(i)}$ in the common space $\boldsymbol{U}_p$.

According to the principle of SVD, the projection axes $\boldsymbol{U}_p$ can be also used to reconstruct a new multivariate time series in the original space taking the form

$$\boldsymbol{Y}_i = \boldsymbol{P}_i\boldsymbol{U}_p^\top = \boldsymbol{X}_T^{(i)}\boldsymbol{U}_p\boldsymbol{U}_p^\top, \ i = 1, \ldots, n, \tag{54}$$

where $\boldsymbol{Y}_i \in \mathbb{R}^{T\times d}$ is the reconstruction of the time series $\boldsymbol{X}_T^{(i)}$ with respect to the projection axes $\boldsymbol{U}_p$. Note that, as the first p principal components are retained, some information may be lost in the process of space transformation so that the reconstruction series might be different from the original one. In other words, some amount of reconstruction error is expected to happen between both series. Formally, the reconstruction error between $\boldsymbol{X}_T^{(i)}$ and $\boldsymbol{Y}_i$ is defined as

$$E_i = \left\| vec\left(\boldsymbol{X}_T^{(i)}\right) - vec(\boldsymbol{Y}_i)\right\|, \ i = 1, \ldots, n. \tag{55}$$

Note that the reconstruction error is caused by two factors, namely the reduced dimensionality of the projection axes and the quality of the common space created through CPCA.

The main idea of Mc2PCA method is to consider the common projection axes associated with a given cluster as a prototype for that group, i.e., the projection axes play the role of the centroid in a *K*-means clustering procedure. In this way, the error between each series and its reconstructed counterpart with respect to the different projection axes can be used to assign each MTS to a cluster. The specific steps concerning the Mc2PCA algorithm are explained below.

1. The original set of MTS, $\boldsymbol{X} = \left\{\boldsymbol{X}_T^{(1)}, \ldots, \boldsymbol{X}_T^{(n)}\right\}$, is randomly divided into K groups, obtaining the set of clusters $\left\{\boldsymbol{C}_1^0, \ldots, \boldsymbol{C}_K^0\right\}$, being each $\boldsymbol{C}_k^0, k = 1, \ldots, K$, approximately formed by the same number of series. The common covariance matrix of $\boldsymbol{X}$ is computed as indicated in (52) and the first p eigenvectors are selected according to a certain criterion.
2. At the iteration gth, the common projection axes of each element in the set $\left\{\boldsymbol{C}_1^g, \ldots, \boldsymbol{C}_K^g\right\}$ is computed, being $\boldsymbol{C}_k^g$ the kth cluster associated with the gth iteration. Specifically, cluster $\boldsymbol{C}_k^g$ gives rise to the projection axes $\boldsymbol{U}_{p,k}^g$, which is obtained by applying SVD to the common covariance matrix computed from the series in the kth cluster. The first p eigenvectors are selected for constructing each $\boldsymbol{U}_{p,k}^g, k = 1, \ldots, K$.
3. For each $\boldsymbol{X}_T^{(i)}$ in the original collection, its reconstructed counterpart with respect to $\boldsymbol{U}_{p,k}^g$ is computed as

$$\boldsymbol{Y}_{i,k}^{g} = \boldsymbol{X}_{T}^{(i)} \boldsymbol{U}_{p,k}^{g} \boldsymbol{U}_{p,k}^{g\top}, \; i = 1, \ldots, n, \; k = 1, \ldots, K, \tag{56}$$

where $\boldsymbol{Y}_{i,k}^{g}$ is the reconstruction of $\boldsymbol{X}_{T}^{(i)}$ with respect to the projection axes $\boldsymbol{U}_{p,k}^{g}$ in the gth iteration.

4. Each series $\boldsymbol{X}_{T}^{(i)}$ is assigned to the cluster $k'(i) = \arg\min_{k=1,\ldots,K} \left\| vec\left(\boldsymbol{X}_{T}^{(i)}\right) - vec\left(\boldsymbol{Y}_{i,k}^{g}\right) \right\|$. The set of clusters is updated accordingly.
5. Steps 2–4 are repeated until no MTS is reassigned or the number of iterations exceeds a certain value, g_m.
6. Assuming that the algorithm stopped at the g_sth iteration, the set of clusters, $\{\boldsymbol{C}_{1}^{g_s}, \ldots, \boldsymbol{C}_{K}^{g_s}\}$ is returned.

The main hyperparameters of Mc2PCA are the number of clusters, K, and the number p of principal eigenvectors. The former can be selected by employing an heuristic criterion based on clustering quality indexes, whereas the latter can be determined by fixing a reasonable rate of desired explained variability. The time complexity of Mc2PCA is $O\left(T\left(nd^2 + g_m nKdp\right) + g_m Kd^3\right)$, indicating that the algorithm could be very slow when tackling datasets of high-dimensional series. Lastly, it is worth emphasizing that the Mc2PCA algorithm was exhaustively analysed by [55] with several real MTS datasets, and the results showed that the proposed technique is superior to various traditional methods for MTS clustering.

A fuzzy clustering algorithm based on VPCA

[52] proposed a fuzzy clustering model based on VPCA and the distance d_{SWMD} defined in (47). The procedure, so-called VPCA$_{\text{FCM}}$, treats each MTS as a reduced matrix and does not employ vectorization. In addition, the spatial dimensionality difference between the entries of the transformed MTS is taken into account. VPCA$_{\text{FCM}}$ performs in a similar way than a traditional fuzzy C-means clustering model. In fact, the corresponding distance between each series and each centroid is computed in terms of d_{SWMD}. Assuming we want to group the set of series $\left\{\boldsymbol{X}_{T}^{(1)}, \ldots, \boldsymbol{X}_{T}^{(n)}\right\}$ into K clusters, the VPCA$_{\text{FCM}}$ clustering algorithm is outlined below (for more details, see Section 3 in [52]):

1. The collection of series $\left\{\boldsymbol{X}_{T}^{(1)}, \ldots, \boldsymbol{X}_{T}^{(n)}\right\}$ is transformed into the collection of reduced fseries $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n\}$ as indicated in (46).
2. The spatial distance matrix $\boldsymbol{S}$ is computed according to (49). K initial centroids are produced from $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n\}$, obtaining the set of centroids $\boldsymbol{Z}^0 = \left\{\boldsymbol{Z}_1^0, \ldots, \boldsymbol{Z}_K^0\right\}$.
3. At the gth iteration, the dissimilarity matrix $\boldsymbol{D}^g = (D_{ki}^g), k = 1, \ldots, k, i = 1, \ldots, n$, where $D_{ki}^g = d_{SWMD}\left(\boldsymbol{Z}_k^g, \boldsymbol{X}_T^{(i)}\right)$, with $\boldsymbol{Z}_k^g$ being the kth centroid associated with the gth iteration, is computed. Note that matrix $\boldsymbol{D}^g$ contains the distances between each series and each centroid.
4. The membership matrix $\boldsymbol{U}^g = (u_{ki}^g), k = 1, \ldots, k, i = 1, \ldots, n$, is computed in the following way:

$$u_{ki}^{g} = \frac{D_{ki}^{g\,\frac{-2}{m-1}}}{\sum_{k=1}^{K} D_{ki}^{g\,\frac{-2}{m-1}}}, \tag{57}$$

where $u_{ki}^g \geqslant 0$ is the membership degree of the ith MTS in cluster kth associated with the gth iteration and $m > 1$ is a parameter which regulates the fuzziness of the partition, usually referred to as the fuzziness parameter. Note that $\sum_{k=1}^{K} u_{ki}^g = 1 \; \forall i \in \{1, \ldots, n\}$, i.e., the sum of the membership degrees associated with a given series must be equal to one, which is a standard constraint in fuzzy clustering algorithms.

5. The set of centroids is updated, obtaining $\boldsymbol{Z}^g = \{\boldsymbol{Z}_1^g, \ldots, \boldsymbol{Z}_K^g\}$, where

$$\boldsymbol{Z}_k^g = \frac{\sum_{i=1}^{n} \boldsymbol{Y}_i u_{ki}^m}{\sum_{i=1}^{n} u_{ki}^m}. \tag{58}$$

6. The value of the objective function is computed as

$$J = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ki}^m D_{ki}^{g\,2}. \tag{59}$$

7. Steps 3–6 are repeated until J is improved less than a certain threshold or the number of iterations exceeds a certain value, g_m.
8. Assuming that the algorithm stopped at the g_sth iteration, the resulting membership matrix $\boldsymbol{U}^{g_s}$ and the resulting set of centroids $\boldsymbol{Z}^{g_s}$, are returned.

Concerning the hyperparameters of the algorithm VPCA$_{\text{FCM}}$, the most important ones are the rate of explained variability employed to construct the reduced matrices $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n\}$, the number of clusters K, and the fuzziness parameter m. Our empirical studies have shown that a rate of explained variability of 0.90 is usually appropriate to get satisfactory results. Regarding, K and m, several heuristic criteria are available for their selection, e.g., criteria based on internal clustering quality indexes.

With respect to the computational time of the method, the approximate time complexity is $O\left(dT^2 n + dT^3\right) + O(g_m Kndp^*)$. The first term is associated with the computation of the transformed series through VPCA (Step 1 before), whereas the second term involves the calculation of the spatial distance matrix and the updating optimization task regarding the fuzzy C-means type procedure (Steps 2–7 before). It is clear that the execution of the algorithm can take a very high time when dealing with long series.

Finally, it is worth remarking that the VPCA$_{\text{FCM}}$ method was empirically assessed by [52] by using publicly available datasets and the results indicated that VPCA$_{\text{FCM}}$ is superior to other procedures in terms of clustering effectiveness and outperforms techniques based on classical PCA with regards to time consumption.

A general forecasting procedure for MTS

Package **mlmts** includes a general technique to predict future values of a given multivariate series. It is worth noting that forecasting is not among the main objectives of the toolbox **mlmts**, particularly due to the availability of well-known R packages to address this problem. However, we believe that adding some specific functions to forecast would be useful for **mlmts** users. For instance, these functions could be of great interest to perform clustering or classification taking point forecasts as features. The proposed method, called Flexible Forecasting for MTS (FFMTS), is based on fitting regression models to lag-embedding matrices of a given autoregressive order. Assuming we want to forecast a given MTS $\boldsymbol{X}_T$ with univariate components $\boldsymbol{X}_T^{U,i} = \left\{X_1^{U,i}, \ldots, X_T^{U,i}\right\}, i = 1, \ldots, d$, the corresponding steps are described below.

1. Fix a maximum lag $L \geqslant 1$, a prediction horizon $h \geqslant 1$ and a standard regression model.
2. For each component $\boldsymbol{X}_T^{U,i}$ in series $\boldsymbol{X}_T, i = 1, \ldots, d$, construct the corresponding lag-embedding matrix as

Table 1
Summary of the 30 datasets contained in the UEA archive. An asterisk indicates that the corresponding database is too large to be included in package **mlmts**.

Dataset	Train instances	Test instances	Dimensions	Length	Classes
ArticularlyWordRecognition	275	300	9	144	25
AtrialFibrillation	15	15	2	640	3
BasicMotions	40	40	6	100	4
CharacterTrajectories	1422	1436	3	182	20
Cricket	108	72	6	1197	12
DuckDuckGeese	60	40	1345	270	5
EigenWorms	128	131	6	17984	5
Epilepsy	137	138	3	206	4
EthanolConcentration	261	263	3	1751	4
ERing	30	30	4	65	6
FaceDetection*	5890	3524	306	375	6
FingerMovements	316	100	28	50	2
HandMovementDirection	160	74	10	400	4
Handwriting	150	850	3	152	26
Heartbeat	204	205	61	405	2
JapaneseVowels	270	370	12	29	9
Libras	180	180	2	45	15
LSST	2459	2466	6	36	14
InsectWingbeat*	30000	20000	200	78	10
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PenDigits	7494	3498	2	8	10
PEMS-SF	267	173	963	144	7
Phoneme	3315	3353	11	217	39
RacketSports	151	152	6	30	4
SelfRegulationSCP1	268	293	6	896	2
SelfRegulationSCP2	200	180	7	1152	2
SpokenArabicDigits	6599	2199	13	93	10
StandWalkJump	12	15	4	2500	3
UWaveGestureLibrary	120	320	3	315	8

$$\boldsymbol{LE}^i = \begin{pmatrix} X_1^{U,1} & \dots & X_L^{U,1} & \dots & X_1^{U,d} & \dots & X_L^{U,d} & X_{L+1}^{U,i} \\ X_2^{U,1} & \dots & X_{L+1}^{U,1} & \dots & X_2^{U,d} & \dots & X_{L+1}^{U,d} & X_{L+2}^{U,i} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ X_{T-L}^{U,1} & \dots & X_{T-1}^{U,1} & \dots & X_{T-L}^{U,d} & \dots & X_{T-1}^{U,d} & X_T^{U,i} \end{pmatrix}.$$

3. For each one of the matrices $\boldsymbol{LE}^i, i = 1, \dots, d$, fit the regression model selected in Step 1 by considering the last column as the response variable and the remaining columns as predictors. This produces a set of d forecasting models, $\mathscr{M} = \{\mathscr{M}_1, \dots, \mathscr{M}_d\}$.
4. Use the set $\mathscr{M}$ to obtain the h-step ahead forecasts for $\boldsymbol{X}_T$ in a recursive manner, obtaining the matrix of predictions

$$\begin{pmatrix} H_1^1 & H_1^2 & \dots & H_1^d \\ H_2^1 & H_2^2 & \dots & H_2^d \\ \dots & \dots & \dots & \dots \\ H_h^1 & H_h^2 & \dots & H_h^d \end{pmatrix}, \tag{60}$$

where H_j^i denotes the j-step ahead forecast for the ith component of $\boldsymbol{X}_T$ according to the model $\mathscr{M}_i$.

The described procedure forecasts each univariate series by considering past values of all the components up to a given lag L, which must be determined by the user. Since any regression model can be fitted to the lag-embedding matrices, the method is quite flexible and allows to construct several classes of forecasting approaches. For instance, a VAR-type prediction function can be obtained by considering the standard linear regression model. Indeed, this approach is expected to yield reliable forecasts if the corresponding MTS exhibits a linear behavior. On the contrary, more complex regression models (random forest, neural networks...) should be selected to get accurate predictions when dealing with series clearly deviating from linearity. Note that the proposed technique is not limited to forecasting, since the prediction matrix obtained in Step 4 could be used as input for alternative machine learning tasks. It is worth highlighting that several approaches relying on lag-embedding matrices have been successfully applied in the field of time series forecasting (see, e.g., [56]).

This way, **mlmts** also includes a simple but powerful forecasting method, enabling users to construct different classes of forecasting functions with several degrees of complexity. The regression models available for Step 1 of the algorithm are implemented by using the R package **caret** [57].

3. Datasets in mlmts

The package **mlmts** includes several MTS datasets which can be used to evaluate different machine learning algorithms, or simply for illustrative purposes. Specifically, **mlmts** contains 28 of the 30 real MTS databases provided in the well-known UEA multivariate time series classification archive [23,37], originally devised in the context of supervised classification, one financial database, and two synthetic datasets. Datasets in the UEA archive cover a wide range of instances, dimensions, series lengths and number of classes. Training and testing sets are established in the archive for each database, thus allowing to perform a rigorous assessment of new methods for MTS classification. Clustering algorithms can also be evaluated with the UEA data collections by considering that the ground truth is determined by the underlying class labels. In fact, the performance of any method assuming certain classes for the time series could be analyzed by means of the UEA databases. Additionally, assessing forecasting procedures using MTS datasets in the UEA archive is also plausible. In sum, by using the UEA data, **mlmts** attempts to provide practitioners in the field of MTS data mining with a simple tool to test their ideas. It is worth highlighting that, to the best of our knowledge, **mlmts** is the first package making available UEA databases to the R community. Table 1 contains a summary of the 30 UEA datasets included in the UEA archive. An asterisk was used to indicate the largest datasets in

the archive, FaceDetection and InsectWingbeat. In both cases, the corresponding files have sizes of over 1 GB, thus making impractical the introduction of these databases in **mlmts**[1]

The synthetic datasets were created so that they contain groups of series generated from the same underlying stochastic process. Specifically, the dataset called SyntheticData1 includes 15 series of length 400 simulated from each one of four different models, thus encompassing a total of 60 MTS. The generating models concerning each class of process are given below.

(a) A VAR(1) process given by

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix} \begin{pmatrix} X_{t-1,1} \\ X_{t-1,2} \end{pmatrix} + \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}.$$

(b) A VMA(1) process given by

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix} \begin{pmatrix} \epsilon_{t-1,1} \\ \epsilon_{t-1,2} \end{pmatrix} + \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}.$$

(c) A QVAR(1) (quantile vector autoregressive) process given by

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \begin{pmatrix} 0 & -0.5(U_{t,1} - 0.5) \\ -0.5(U_{t,2} - 0.5) & 0 \end{pmatrix} \begin{pmatrix} X_{t-1,1} \\ X_{t-1,2} \end{pmatrix} + \begin{pmatrix} \Phi^{-1}(U_{t,1}) \\ \Phi^{-1}(U_{t,2}) \end{pmatrix}.$$

(d) A QVAR(1) process given by

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \begin{pmatrix} 0 & 1.5(U_{t,1} - 0.5) \\ 1.5(U_{t,2} - 0.5) & 0 \end{pmatrix} \begin{pmatrix} X_{t-1,1} \\ X_{t-1,2} \end{pmatrix} + \begin{pmatrix} \Phi^{-1}(U_{t,1}) \\ \Phi^{-1}(U_{t,2}) \end{pmatrix}.$$

In processes (a) and (b), $(\epsilon_{t,1}, \epsilon_{t,2})^\top$ is an i.i.d. vector error process following the bivariate standard normal distribution. In processes (c) and (d), $= (U_{t,1}, U_{t,2})^\top$ is a sequence of independent random vectors with independent components $U_{t,k}$ which are uniformly distributed on $[0,1]$, and $\Phi^{-1}(u), u \in (0,1)$, denotes the quantile function of the standard normal distribution.

Note that, in SyntheticData1, each underlying model can be seen as a different class label in the context of supervised learning. Moreover, these underlying models can be regarded as the ground truth in the case of unsupervised learning. Therefore, SyntheticData1 is an appropriate collection to illustrate the use of methods assuming that the different classes/groups are given by different dependence structures. An interesting feature of this collection is that it combined linear processes with highly nonlinear models.

In order to create a database containing some anomalous series, the set SyntheticData1 was expanded by incorporating 5 MTS generated from the VAR(1) model

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \begin{pmatrix} -0.4 & -0.4 \\ 0.4 & 0.4 \end{pmatrix} \begin{pmatrix} X_{t-1,1} \\ X_{t-1,2} \end{pmatrix} + \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix},$$

giving rise to SyntheticData2. Note that this latter collection is useful to demonstrate the use of **mlmts** for anomaly detection purposes.

Finally, the financial dataset included in **mlmts** is called FinancialData and contains 50 series representing companies of the well-known S&P500 index. This data collection is analysed in detail in Section 4.4.

4. Using the mlmts package. An illustration

In this section, we illustrate the use of **mlmts** by employing several implemented functions with some UEA databases and also a synthetic dataset included in the own package. Notice that the examples involving real data do not intend to reach meaningful conclusions about the underlying problems, but rather to show the scope and usefulness of the package **mlmts**.

As explained in previous sections, **mlmts** provides several functions to calculate dissimilarity measures between multivariate series, as well as some methods implementing particular machine learning procedures. In order to get familiar with the commands of package **mlmts**, Table 2 lists each of the distance measures introduced in Section 2 together with the corresponding **mlmts** functions. In the same way, Table 3 associates each one of the specific data mining methods with the corresponding function in the package. After a brief introduction to the general framework of **mlmts**, the rest of the section illustrates the use of the package separately for the cases of classification, clustering, outlier detection, dimensionality reduction and forecasting.

4.1. Some generalities about mlmts

In **mlmts**, a T-length, d-dimensional MTS $\boldsymbol{X}_T \in \mathbb{R}^{T\times d}$ is represented through a matrix object with T rows and d columns. In this way, each column of a MTS object can be regarded as a vector containing a T-length UTS.

The majority of functions in the package take as input a list of MTS (i.e., a list of matrices). Functions in Table 2 return by default a pairwise dissimilarity matrix (*dist* object) based on the corresponding distance measure. Most of these functions admit the argument *features = TRUE*. In that case, the function returns a dataset in which the ith row corresponds to a feature vector associated with the ith MTS in the input list. The nature of the feature vectors depends on the underlying dissimilarity measure. For instance, the function *dis_cor()* produces by default a pairwise dissimilarity matrix according to (13). However, if we employ the argument *features = TRUE*, a dataset whose ith row contains a vector of estimated auto and cross-correlations for the ith MTS (i.e., a vector including the quantities $\widehat{\rho}_{j_1 j_2}(l)$ in (13)), is returned. In this manner, functions in Table 2 are not limited to the computation of distances. On the other hand, the output of functions in Table 3 does not follow a common pattern, being related to the specific context. It is worth remarking that all functions in **mlmts** require the input series to be numerical and have the same number of dimensions. On the contrary, the requirement on the equality of lengths depends on the particular function.

The UEA databases, the synthetic datasets and the financial data collection included in **mlmts** are defined by means of a list named as indicated in the first column of Table 1. Each list contains two elements, which are described below.

- The element called *data* is a list of matrices with the multivariate series of the corresponding collection.
- The element named *classes* includes a vector of class labels associated with the objects in *data*. The only exception is the financial dataset. In this case, as no underlying class labels exist, *classes* produces the abbreviated names of the companies as given in the S&P500 index (see Section 4.4).

MTS objects can be easily visualized in **mlmts** by using the function *mts_plot()*, which takes as input a numerical matrix. For instance, the following code can be used to visualize the first series in RacketSports and SyntheticData1 datasets, respectively:

```
>library(mlmts)
>mts_plot(RacketSports$data[[1]])
>mts_plot(SyntheticData1$data[[1]])
```

[1] Datasets FaceDetection and InsectWingbeat can be downloaded from https://www.timeseriesclassification.com/, where additional information about the UEA archive is available.

Table 2
Dissimilarity measures implemented in **mlmts**.

Distance measure	Function in **mlmts**
Shape-based	
d_{EUCL}	*dis_eucl()*
d_F	*dis_frechet()*
d_{DTW1}	*dis_dtw1()*
d_{DTW2}	*dis_dtw2()*
d^*_{MD}	*dis_mahalanobis()*
d_{MDDTW}	*dis_dtw_mahalanobis()*
Feature-based	
d_{COR}	*dis_cor()*
d_{GCC}	*dis_gcc()*
d_{JSPEC}	*dis_spectral(method = 'j_divergence')*
d_{CSPEC}	*dis_spectral(method = 'chernoff_divergence')*
d_{MODWT}	*dis_modwt()*
d_{QCF}	*dis_qcf()*
d_{QCD}	*dis_qcd()*
d_{WWW}	*dis_www()*
d_{HWL}	*dis_hwl()*
$d_{ZAGORECKI}$	*dis_zagorecki()*
Model-based	
d_{VAR}	*dis_var_1()*
$d_{VAR,p}$	*dis_var_2()*
Based on dimensionality reduction	
d_{Eros}	*dis_eros()*
d_{PCA}	*dis_pca()*
d_{2dSVD}	*dis_2dsvd()*
d_{LPP}	*dis_lpp()*
d_{SWMD}	*dis_swmd()*
d_{PPCA}	*dis_ppca()*
d_{MCC}	*dis_mcc()*

Table 3
Specific machine learning procedures implemented in **mlmts**.

Specific procedure	Context	Function in **mlmts**
F4	Classification	*f4_classifier()*
QCD-F-OD	Outlier detection	*outlier_detection()*
Mc2PCA	Clustering	*mc2pca_clustering()*
$VPCA_{FCM}$	Clustering	*vpca_clustering()*
FFMTS	Forecasting	*mts_forecasting()*

The corresponding graphs are depicted in Fig. 3. The function *mts_plot()* represents a MTS by showing each dimension (UTS) separately, with a different color, thus providing an overall picture of the behavior of the series. According to the top panel of Fig. 3, the first series in RacketSports is formed by dimensions exhibiting different types of behaviour. On the other hand, no trend patterns are observed for the MTS in the bottom panel, which is expected since the series was generated from a stationary VAR process. In sum, the function *mts_plot()* provides an useful tool to perform exploratory analysis of MTS objects.

4.2. Performing MTS classification with mlmts

The package **mlmts** is a great tool to perform supervised classification of MTS. Firstly, we show how the dissimilarities associated with the functions in Table 2 can be used to construct a *k*NN classifier. We start by testing several dissimilarities with the dataset BasicMotions. For exploratory purposes, we have depicted in Fig. 4 four MTS of this database, each one of them pertaining to one of the four different classes. It is clear from Fig. 4 that each class is characterized by a different geometric pattern. For instance, series in Classes 2 and 4 take much larger values than series in

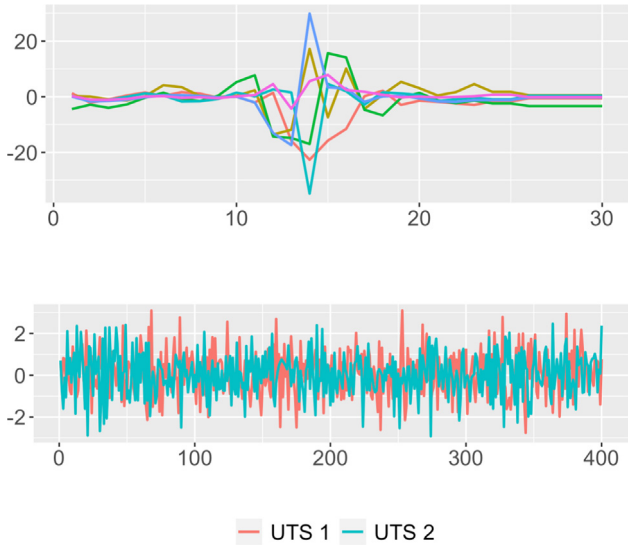


Fig. 3. First MTS in the RacketSports (top panel) and SyntheticData1 (bottom panel) datasets.

Classes 1 and 3. This suggests that a shape-based dissimilarity could be effective in classifying the series of dataset BasicMotions.

Function *knn_classifier()* in **mlmts** is specifically designed to compute *k*NN classification. Below we illustrate the use of *knn_classifier()* by considering the SVD-based dissimilarity d_{Eros}.

```
>predictions <- knn_classifier(dataset
  = BasicMotions$data, classes=BasicMotions
  $classes, index_test = 41:80, distance =
  'dis_eros', k = 1)
```

The input parameters of *knn_classifier()* are the MTS dataset (*dataset*), the associated vector of classes (*classes*), the indexes of the elements defining the test set (*index_test*), the distance function to be employed, as a string (*distance*), and the number of neighbors (*k*). Note that, in this example, the last 40 elements in BasicMotions are selected as test set by following Table 1. The function returns the predictions (class labels) for the elements in the test set, which are stored in the vector *predictions*. To evaluate the performance of the algorithm, we can compute the classification accuracy by comparing the predicted classes with the true labels. Such computation can be carried out by means of the function *ml_test()* of package **mltest** [58].

```
>library(mltest)
>ml_test(predictions, BasicMotions$classes
  [41:80])$accuracy
[1] 0.7
```

Thus, the *k*NN classifier based on the distance d_{Eros} achieves an accuracy of 0.70, showing substantially better performance than a naive classifier, which has an expected accuracy of 0.25 (both training and test sets in BasicMotions are balanced). The use of alternative metrics can be tested in a similar way. Consider for instance the dissimilarity based on the MODWT. We evaluate this distance for $k = 1$ and $k = 5$ in order to analyze the effect of the number of neighbors.

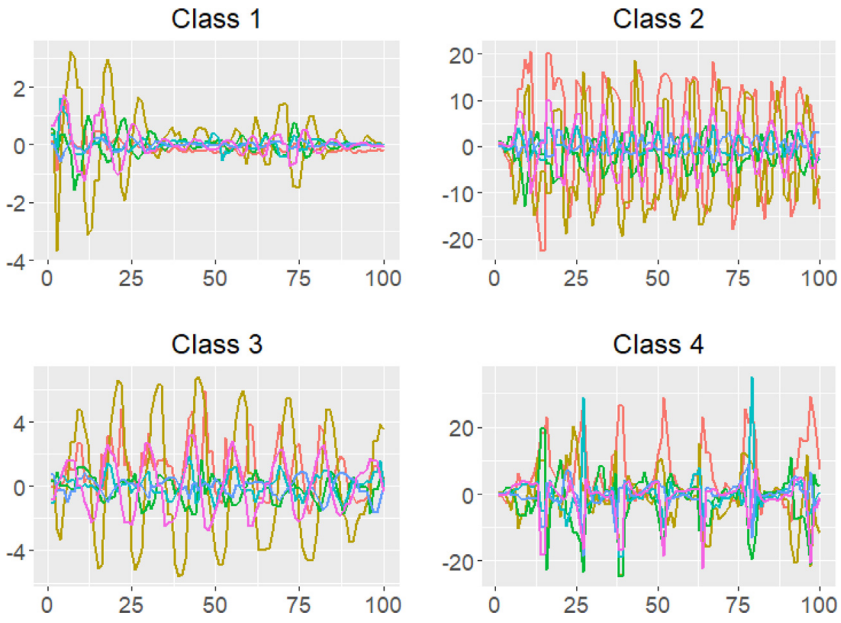


Fig. 4. Four MTS in dataset BasicMotions. Each series pertains to one of the four different classes.

```
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_modwt', k = 1)
>ml_test(predictions, BasicMotions$classes[41:
  80])$accuracy
[1] 1
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_modwt', k = 5)
>ml_test(predictions, BasicMotions$classes[41:
  80])$accuracy
[1] 1
```

```
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_2dsvd', k = 1)
>ml_test(factor(predictions, levels = c(1, 2, 3,
  4)), BasicMotions$classes[41:80])$ccuracy
[1] 0.55
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_lpp', k = 1)
>ml_test(predictions, BasicMotions$classes[41:
  80])$accuracy
[1] 0.65
```

In both cases, the metric d_{MODWT} attains perfect results. Let's now examine the behavior of two geometric distances, namely d_{EUCL} and d_{DTW1}.

```
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_eucl', k = 1)
>ml_test(factor(predictions, levels = c(1, 2, 3,
  4)), BasicMotions$classes[41:80])$accuracy
[1] 0.6
>predictions <- knn_classifier(dataset
  = BasicMotions$data, BasicMotions$classes,
index_test = 41:80, distance = 'dis_dtw_1', k = 1)
>ml_test(predictions, BasicMotions$classes[41:
  80])$accuracy
[1] 0.95
```

Therefore, even though the differences in shape are substantial, the Euclidean distance provides worse performance than the previous dissimilarities, whereas d_{DTW1} exhibits a great effectiveness. Finally, we assess the distances d_{2dSVD} and d_{LPP}, which rely on SVD.

Both metrics lead to results far worse than the ones reached by the best-performing dissimilarities, thus indicating that some amount of information is lost during the dimensionality reduction process. In sum, the best metrics to classify series in BasicMotions are d_{MODWT} and d_{DTW1}. Note that the great performance of the latter was expected from the analysis of Fig. 4.

The nearest neighbors rule is one of the several options available in **mlmts** to perform supervised classification. An alternative approach consists of using the feature dataset provided by most of the functions in Table 2 to feed a traditional classifier. Next we illustrate this approach by considering the data collection SyntheticData1. The corresponding feature dataset associated with the distance d_{QCF} is obtained as follows.

```
>feature_dataset <- dis_qcf(SyntheticData1$data,
  features = T)
```

The object *feature_dataset* is a matrix whose *i*th row contains the vector of quantile-based estimates describing the *i* th MTS. In fact, the Euclidean distance between the *i*th and *j* th rows of this matrix is the dissimilarity d_{QCF} between the series *i and j*. Many traditional classification algorithms can be applied to the object *feature_dataset* by means of the R package **caret**. As the SyntheticData1 does not have a default split into training and test

set, the evaluation of the algorithms in this database can be carried out, for instance, by using Leave-One-Out Cross-Validation (LOOCV). Package **caret** requires the dataset of features to be an object of class *data.frame* whose last column provides the class labels of the elements, which must be named *'Class'*. Based on previous comments, the following chunk of code creates the object *df_feature_dataset*, which can be used as input to **caret** functions.

```
>feature_dataset <- dis_qcf(SyntheticDatal$data,
  features = T)
>df_feature_dataset <- data.frame(cbind
  (feature_dataset,
SyntheticDatal$classes))
>d <- dim(df_feature_dataset)[2]
>colnames(df_feature_dataset)[d] <- 'Class'
>df_feature_dataset[,d] <- factor
  (df_feature_dataset[,d])
```

The function *train()* can be used to fit several classifiers to the feature dataset. A grid search in the hyperparameter space of the corresponding classifier is performed by default. First we consider a random forest by using *method = 'ranger'* as input parameter. By means of the argument *trControl*, we define LOOCV as evaluation protocol.

```
>library(caret)
>train_control <- trainControl(method = 'LOOCV')
>model <- train(Class ., data = df_feature_dataset,
  trControl = train_control, method = 'ranger')
```

The object *model* contains the fitted model and the evaluation results, among others. The corresponding accuracy can be accessed as follows.

```
>max(model$results$Accuracy)
[1] 1
```

The QCD-based algorithm achieves maximum accuracy when classifying the series in SyntheticDataset1, thus perfectly discriminating between the four underlying stochastic processes.

Next we study the performance of the correlation-based features based on d_{COR}.

```
>feature_dataset <- dis_cor(SyntheticDatal$data,
  features = T)
>df_feature_dataset <- data.frame(cbind
  (feature_dataset, SyntheticDatal$classes))
>d <- dim(df_feature_dataset)[2]
>colnames(df_feature_dataset)[d] <- 'Class'
>df_feature_dataset[,d] <- factor
  (df_feature_dataset[,d])
>model <- train(Class ., data = df_feature_dataset,
  trControl = train_control, method = 'ranger')
>max(model$results$Accuracy)
[1] 0.8166667
```

The metric d_{COR} attains moderately worse results than d_{QCF}, with a maximum accuracy of 0.817 for the best combination of hyperparameters in the random forest. Let's try now a different classifier, for instance, the standard Quadratic Discriminant Analysis (QDA) (*method='qda'*).

```
>model <- train(Class ., data = df_feature_dataset,
trControl = train_control, method = 'qda')
>max(model$results$Accuracy)
[1] 0.7166667
```

The results attained with QDA are poorer, with a maximum accuracy of 0.717. The worse performance of d_{COR} with respect to d_{QCF} in classifying the series of SyntheticData1 is caused by the fact that the QVAR processes defining the last two classes in this dataset have all theoretical auto and cross-correlations equal to zero. Therefore, the series generated from both QVAR models are indistinguishable from the point of view of the correlation-based quantities, resulting in several misclassifications. On the contrary, the quantile-based estimates are able to detect arbitrary dependence patterns as the ones arising in both QVAR processes, which endows the corresponding classifier with a great ability to differentiate between complex stochastic structures.

Still on the classification task, **mlmts** includes an additional specific function implementing the algorithm F4 proposed by [17], so-called *f4_classifier()*. To illustrate the use of this command, we are going to perform supervised classification with the dataset Libras. We want to fit the classifier with respect to the training set (first 180 elements in Libras) and to produce predictions for the elements in the test set (last 180 elements in Libras). The function *f4_classifier()* provides a simple way to execute this process in a single line of code.

```
>predictions_libras <- f4_classifier
  (training_data = Libras$data[1:180],
new_data = Libras$data[181:360], classes = Libras
  $classes[1:180])
>ml_test(predictions_libras, Libras$classes[181:
  360])$accuracy
[1] 0.85
```

The object *predictions_libras* is a vector containing the predicted class labels for the MTS in the test set. The corresponding classification accuracy is 0.85, thus indicating that classifier F4 attains great results in a quite challenging problem as the one defined by the Libras dataset, where 15 different classes are present. Note that the argument *new_data* in the function *f4_classifier()* is not limited to be a test set. In fact, it can be any collection of MTS for which the user wishes to obtain the predicted labels. The only condition is that the objects in *new_data* must have the same structure (number of rows and columns) than the objects in *training_data*.

It is worth remarking that *f4_classifier()* can be executed without providing data requiring predictions. Actually, *new_data = NULL* constitutes the default option for the command. In that case, the function returns a fitted model, which is an object of class *train* of the **caret** package. Let's exemplify this scenario by considering the dataset RacketSports. For illustrative purposes, training and test sets are used in conjunction to fit the classifier.

```
>model <- f4_classifier(training_data
  = RacketSports$data, classes = RacketSports
  $classes)
>max(model$results$Accuracy)
[1] 0.8383994
```

The accuracy obtained via cross-validation is approximately 0.84.

4.3. Performing MTS clustering with mlmts

The package **mlmts** also provides an excellent framework for practitioners to carry out MTS clustering. In this context, we consider the dataset SyntheticData1. Assuming that the clustering structure is governed by the similarity between underlying models, the ground truth is given by the 4 groups involving the 15 series from the same generating process. Thus, this partition has to be compared with the experimental solutions to assess clustering algorithms.

We start testing the dissimilarity d_{JSPEC} proposed by [4], which measures discrepancy between MTS by taking into account estimates of the spectral density matrices. To compute a pairwise dissimilarity matrix based on d_{JSPEC}, we feed the function *dis_spectral()* with the corresponding data object, indicating *method = 'j_divergence'*.

```
>distance_matrix_jspec <- dis_spectral(SyntheticData1$data,
method = 'j_divergence')
```

The object *distance_matrix* is a *dist* object which can be taken as input for several standard clustering algorithms. It contains the dissimilarity matrix calculated by applying d_{JSPEC} to each pair of series in *SyntheticData1$data*. As the correct number of clusters is known', we first consider a partitive clustering technique as the popular Partitioning Around Medoids (PAM) algorithm. We employ the function *pam()* of package **cluster** [59].

```
>library(cluster)
>spectral_clustering_pam <- pam
  (distance_matrix_jspec, k = 4)$clustering
>spectral_clustering_pam
[1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  3  3  3  3 [35]  3  3  3  4  3  3
  3  3  4  4  3  4  4  4  4  4  3  4  4  4
  4  4  4  4  4  4
```

The vector *spectral_clustering_pam* provides the experimental partition. From this solution we can conclude that the series of the first two processes (the linear ones) get correctly assigned, whereas d_{JSPEC} has some trouble in discriminating series from the last two models (the QVAR ones). As measure of clustering effectiveness, we consider the so-called Adjusted Rand Index (ARI) [60]. This index can be easily computed by means of the function *external_validation()* of package **ClusterR** [61].

```
>ground_truth <- rep(1:4, each = 15)
>external_validation(ground_truth,
  spectral_clustering_pam)
[1] 0.8382027
```

The ARI index is bounded between −1 and 1 and admits a simple interpretation: the closer it is to 1, the better is the agreement between the ground truth and the experimental solution. Moreover, the value of 0 is associated with a clustering partition picked at random according to some simple hypotheses. Therefore, it can be concluded that the d_{JSPEC} attains good results in this scenario when used with the PAM algorithm. Note that the high value of ARI index was already expected from the output of *spectral_clustering_pam*.

The popular *K*-means clustering algorithm can be also executed by using **mlmts** utilities. In this case, we need to employ a dataset of features along with the *kmeans()* function of package **stats** [27].

```
>feature_dataset_jspec <- dis_spectral
  (SyntheticData1$data, method = 'j_divergence',
features = T)
>spectral_clustering_kmeans <- kmeans
  (feature_dataset_jspec, centers = 4)$cluster
>external_validation(ground_truth,
  spectral_clustering_kmeans)
[1]  0.8397744
```

In this example, the dissimilarity d_{JSPEC} achieves slightly better results with the *K*-means algorithm, although the difference does not seem statistically significant. Any dissimilarity function can be analyzed in a similar manner. Consider for example the dissimilarity based on fitted VAR coefficients d_{VAR}.

```
>distance_matrix_var <- dis_var_1(SyntheticData1
  $data)
>var_clustering_pam <- pam(distance_matrix_var, k
  = 4)$clustering
>external_validation(ground_truth,
  var_clustering_pam)
 [1] 0.7071614
>feature_dataset_var <- dis_var_1(SyntheticData1
  $data, features = T)
>var_clustering_kmeans <- kmeans
  (feature_dataset_var, centers = 4)$cluster
>external_validation(ground_truth,
  var_clustering_kmeans)
[1] 0.6553356
>var_clustering_pam
[1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  3  3  4  4
[35]  4  3  4  3  4  4  3  3  3  3  3  4  4
  3  3  4  4  3  4  4  4  4  4  4  4  4
>var_clustering_kmeans
[1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  3  2  2  2
[35]  2  3  2  3  2  2  3  3  3  3  3  2  2
  2  3  2  3  3  3  3  2  2  2  2  3  2
```

The VAR-based metric leads to worse results. Here the PAM algorithm outperforms the *K*-means algorithm. The experimental partitions show that d_{VAR} perfectly identifies the series generated from both linear models, but suffers from model misspecification when the generating processes depart from the linearity assumption, thus struggling to differentiate between series simulated from both QVAR models.

A classical exploratory step in cluster analysis consists of constructing the so-called two-dimensional scaling (2DS) based on the pairwise dissimilarity matrix. In short, 2DS represents the pairwise distances in terms of Euclidean distances into a 2-dimensional space preserving the original values as well as possible (by minimizing a loss function). Package **mlmts** includes a function returning an attractive simple representation of a 2DS plane. Below we obtain the 2DS plane associated with the dissimilarity d_{JSPEC}.

```
>plot_2d_scaling(distance_matrix
  = distance_matrix_jspec, cluster_labels
  = ground_truth)
$plot
$gof
[1] 0.6278975
```

The corresponding graph is shown in the top panel of Fig. 5. The points are colored according to the underlying ground truth, which is indicated to function *plot_2d_scaling()* by means of the argument *cluster_labels*. This option is often useful to get insight about whether a specific dissimilarity measure is appropriate in a particular problem when the true class labels are known. This way, **mlmts** provides the user with a simple exploratory tool to assess the adequacy of several metrics in a given MTS database. The top panel of Fig. 5 reveals that the dissimilarity d_{JSPEC} clearly identifies the series from both VAR models (Clusters 1 and 2), but mixes the series from both QVAR processes to some extent. The function *plot_2d_scaling()* also reports the R^2 value (0.63 in this case), which measures the proportion of variance of the original data accounted for the 2DS procedure, thus indicating the quality of the embedding. For comparison purposes, the bottom panel of Fig. 5 displays the 2DS plane corresponding to distance d_{QCD}, which is capable of totally discern the underlying clustering structure in this problem.

To conclude, we show how the algorithm VPCA_{FCM} can be executed through package **mlmts**. We illustrate this algorithm by performing fuzzy clustering in the dataset AtrialFibrillation. To this aim, the object *AtrialFibrillation$data* must be used as input for the function *vpca_clustering()*. We consider the existence of 3 underlying clusters (the number of different classes in the database AtrialFibrillation) and the value 1.5 for the fuzziness parameter (standard choice in the fuzzy clustering literature).

```
>fuzzy_clustering <- vpca_clustering(AtrialFibrillation$data, k = 3, m = 1.5)
>fuzzy_clustering$U[, 1:5]
      [,1]        [,2]        [,3]       [,4]        [,5]
[1,] 0.151486421 0.090469821 0.3587131 0.04954507 0.064394802
[2,] 0.008914912 0.002475682 0.2814761 0.86719024 0.004663519
[3,] 0.839598667 0.907054497 0.3598108 0.08326469 0.930941680
```

The previous output shows the first 5 columns of the resulting membership matrix, i.e., the fuzzy clustering partition for the first 5 MTS in AtrialFibrillation. MTS 1, 2 and 5 get located in the third cluster with membership degrees greater than 0.80 (i.e., $u_{3i} > 0.80, i \in \{1,2,5\}$). On the contrary, MTS 4 is mainly placed in the second group, whereas MTS 3 displays a quite fuzzy allocation, with membership degrees higher than 0.28 in the three clusters. Note that the five analyzed MTS pertain to the same group according to the assumed ground truth. Function *vpca_clustering()* also produces the final set of centroids (Z^{g_s} in Step 8 of VPCA_{FCM} algorithm), which can be seen as prototypes for each one of the clusters. Below we show the centroid of the first cluster.

```
>fuzzy_clustering$centroids[[1]]
       [,1]         [,2]
[1,] 0.298206663  -0.05559328
[2,] -0.345047068  0.92699629
[3,] -0.718630543  0.38115124
[4,] -0.009589561  0.60010414
[5,] 0.429001360  -0.30280082
[6,] 0.644505031  -0.02695864
[7,] 0.639802093  -0.16183344
[8,] 0.878216726  0.17991764
[9,] -0.431324783  0.36750956
[10,] -0.960149246  -0.25486112
```

4.4. Performing outlier detection in MTS datasets with mlmts

Although clustering and classification constitute perhaps two of the most extended approaches in MTS data mining, the package **mlmts** is not limited to such types of algorithms. To confirm the previous claim, this section is devoted to illustrate the use of **mlmts** for an alternative machine learning purpose which is the detection of anomalous series. We should remark that, although there are several definitions of outliers in the context of temporal data (i.e., additive outliers, innovative outliers, etc), **mlmts** is designed to perform outlier detection in such a way that the anomalous series are whole MTS objects. In other words, the anomaly identification task consists of determining some MTS in the dataset whose behavior greatly differs from that of the majority.

To begin with, we show how the dissimilarity matrix produced by the functions in Table 2 can be used itself as a tool to carry out anomaly detection. To this aim, the distance d_{SWMD} and the dataset SyntheticData2 are considered. Note that, as indicated in Section 3, the last 5 series in this database (with indexes from 61 to 65) are anomalous elements since they were deliberately generated from a process different from the ones characterizing the four underlying clusters. Given a distance matrix, the outlier detection task can be carried out in two steps.

1. For each element, computing the sum of the distances between that element and the remaining objects in the dataset, which is expected to be large for anomalous elements.
2. Sorting the quantities computed in Step 1 in decreasing order, then recording the associated vector of indexes. The first indexes in this vector correspond to the most outlying elements and the last indexes in this vector correspond to the least outlying elements.

Following this approach, the top 5 anomalous series in dataset SyntheticData2 according to distance d_{SWMD} can be obtained as follows.

```
>distance_matrix_swmd <- dis_swmd(SyntheticData2
  $data)
>order(colSums(as.matrix(distance_ma-
  trix_swmd)), decreasing = T)[1:5]
[1]  31  65  33  61  64
```

Distance d_{SWMD} is able to identify 3 true outliers series (MTS 61, 64 and 65) but it also misclassifies two nonanomalous MTS as outlying ones (MTS 31 and 33). Let's study now the performance of dissimilarities d_{LPP} and d_{PCA}.

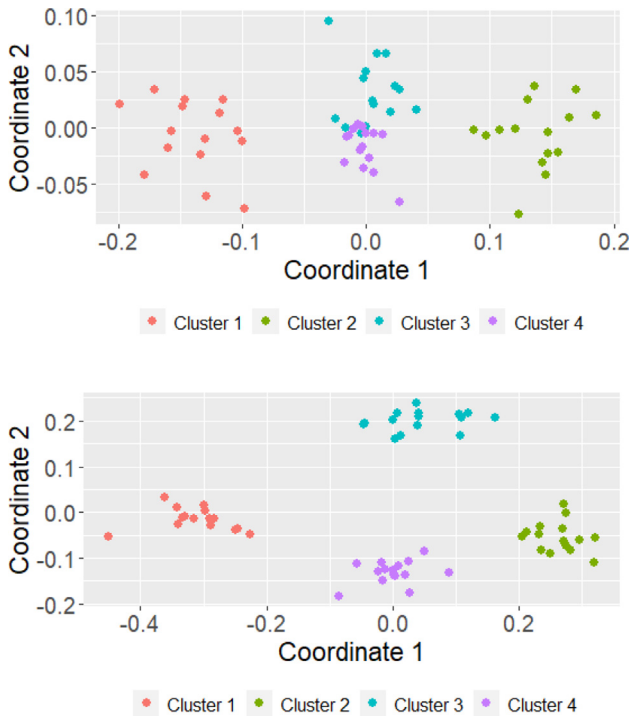


Fig. 5. Two-dimensional scaling planes based on distances d_{JSPEC} (top panel) and d_{QCD} (bottom panel) for the series in the dataset SyntheticData1.

```
>distance_matrix_lpp <- dis_lpp(SyntheticData2
  $data)
>order(colSums(as.matrix(distance_matrix_lpp)),
  decreasing = T)[1:5]
[1]  51  23  61  19  10
>distance_matrix_pca <- dis_pca(SyntheticData2
  $data, retained_components = 2)
>order(colSums(as.matrix(distance_matrix_pca)),
  decreasing = T)[1:5]
[1]  38  52  46  64  34
```

Both metrics detect only one anomalous series, thus exhibiting a poor performance in the corresponding outlier identification task. These results indicate that the dimensionality reduction approach employed by these distances is not effective in detecting series generated from an atypical stochastic process.

Apart from performing outlier identification starting from a pairwise dissimilarity matrix, package **mlmts** also includes the anomaly detection technique proposed by [26], QCD-F-OD, which is based on the treatment of the QCD-features as functional data. The command *outlier_detection()* allows to execute that approach in a simple way.

```
>outlier_detection(SyntheticData2$data)$Indexes
[1]  61  62  64  65  63   9  11  20  14  10
  4  12  13   3  35  53  34  15   7   5  28  18
[23]  1  38  31  21  43  29  32  24   6  40
  45  39  44   2  33  17  25  19  36  37  23
  41  27
[46]  16  59  51  52  42  26  50  56  49  22
  54  60   8  55  47  30  46  57  58  48
```

Note that the function *outlier_detection()* returns a vector including the indexes of the MTS in the corresponding dataset, which are sorted in decreasing order in accordance to the outlying likelihood of the underlying elements. In this example, algorithm QCD-F-OD correctly identifies the 5 true atypical series, thus displaying maximum accuracy.

Finally, our last example involves anomaly detection in a real MTS dataset containing financial time series. The database, so-called FinancialData, includes daily stock returns and trading volume of the top 50 companies of the S&P 500 index according to market capitalization. The sample period spans from 6th July 2015 to 7th February 2018, thus resulting serial realizations of length $T = 655$. The S&P 500 is a stock market index that tracks the stocks of 500 large-cap U.S. companies. The top 50 contains some of the most important companies in the world, as Apple, Google, Facebook or Berkshire Hathaway. The corresponding MTS collection is included in **mlmts** through the list *FinancialData*$*data*. It is worth remarking that the data in FinancialData was analyzed in [26] to study the behaviour of their proposed algorithm.

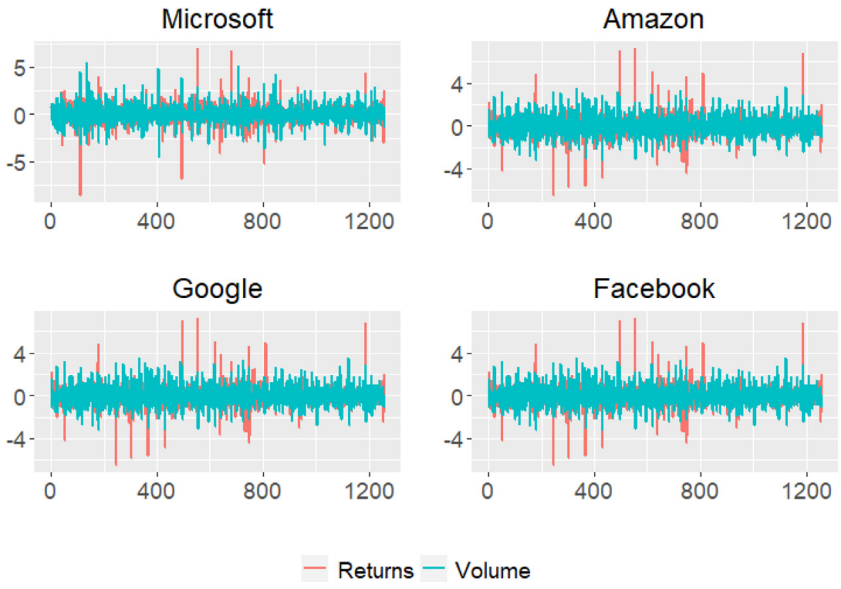


Fig. 6. Returns and change in volume of four American companies in the S&P 500 index.

It can be observed that both the UTS of prices and trading volume are non-stationary in mean. Thus, all UTS are transformed by taking the first differences of the natural logarithm of the original values. This way, prices give rise to stock returns, and volume to what we call change in volume. This kind of transformation is common when dealing with financial time series. Fig. 6 shows four of the transformed bivariate time series, corresponding to companies Microsoft, Amazon, Google and Facebook. All series seem stationary.

Let's assume that up to 10% of the most distant MTS are candidate to exhibit an anomalous behaviour, i.e., our concern is to identify the top 5 extreme series. By treating with real data, a comprehensive analysis of the five identified MTS should be carried out before concluding their outlying performance. We consider several techniques to carry out the outlier identification task. First we consider the algorithm QCD-F-OD through function *outlier_detection()*, whose input parameter *alpha* determines the desired proportion of outliers to detect. Next we execute a distance matrix-based approach relying on dissimilarities $d_{COR}, d_{MODWT}, d_{Eros}$ and d_{DTW1}. Note that vector *FinancialData$classes* contains the abbreviations of the corresponding companies as given in the S&P 500 index.

```
>indexes_outliers_1 <- outlier_detection
  (FinancialData$data, alpha = 0.10)
>FinancialData$classes[indexes_outliers_1]
[1] "TSLA" "NEE" "CVX" "NFLX" "MA"

>distance_matrix_cor <- dis_cor(FinancialData
  $data)
>indexes_outliers_2 <- order(colSums(as.matrix
  (distance_matrix_cor)), decreasing = T)[1: 5]
>FinancialData$classes[indexes_outliers_2]
  [1] "NFLX" "MRK" "BRK.B" "NVDA" "GOOGL"

>distance_matrix_modwt <- dis_modwt
  (FinancialData$data)
>indexes_outliers_3 <- order(colSums(as.matrix
  (distance_matrix_modwt)), decreasing = T)[1: 5]
>FinancialData$classes[indexes_outliers_3]
  [1] "NVDA" "NFLX" "MRK" "XOM" "FB"

>distance_matrix_eros <- dis_eros(FinancialData
  $data)
>indexes_outliers_4 <- order(colSums(as.matrix
  (distance_matrix_eros)), decreasing = T)[1: 5]
>FinancialData$classes[indexes_outliers_4]
  [1] "GOOGL" "NVDA" "NFLX" "MRK" "CVX"

>distance_matrix_dwt_1 <- dis_dtw_1
  (FinancialData$data)
>indexes_outliers_5 <- order(colSums(as.matrix
  (distance_matrix_dwt_1)), decreasing = T)[1: 5]
>FinancialData$classes[indexes_outliers_5]
  [1] "NVDA" "CVX" "QCOM" "LLY" "XOM"
```

Table 4
The 5 anomalous companies according to different approaches.

Function	Anomalous companies
outlier_detection()	Tesla, NextEra Energy, Chevron, Netflix and Mastercard
dis_cor()	Netflix, Merck, Berkshire Hathaway, Nvidia and Google (Class A)
dis_modwt()	Nvidia, Netflix, Merck, ExxonMobil and Facebook
dis_eros()	Google (Class A), Nvidia, Netflix, Merck and Chevron
dis_dtw_1()	Nvidia, Chevron, Qualcomm, Eli Lilly and ExxonMobil

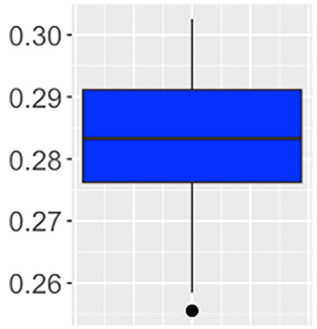


Fig. 7. Boxplots of the depths for the QCD-F-OD procedure applied to the dataset FinancialData.

Each of the approaches gives rise to a different set of anomalous series. Table 4 summarizes the results from the previous outputs. Distances d_{COR}, d_{MODWT} and d_{Eros} identify three common atypical companies, namely Netflix, Merck and Nvidia. On the contrary, the outlier sets detected by QCD-F-OD and d_{DTW1} are more diverse, with companies as Tesla, NextEra Energy, Qualcomm and Eli Lilly, which are not considered anomalous by the remaining approaches. The previous analysis suggests that the conclusions reached from the outlier identification procedure are highly dependent on the chosen dissimilarity measure and its nature. For instance, d_{COR} detects atypical series in terms of linear structures, whereas QCD-F-OD involves general dependence patterns and d_{DTW1} assesses outlyingness in terms of shape. Note that the last dissimilarity does not seem very appropriate to perform outlier identification in dataset FinancialData since the corresponding series are stationary, which makes unlikely that discrepancies between them are due to different geometric profiles. In sum, package **mlmts** provides several dissimilarities to carry out MTS data mining but it is ultimately the user who should decide what is the most appropriate metric for the problem at hand.

Note that, in the previous case study, the number of outliers to detect was set in advance without following any specific principle. However, there exist some heuristic criteria to determine the number of true outliers in the database. One typical rule consists of analyzing the distribution of the quantities determining the outlying likelihood. For instance, in the case of QCD-F-OD, we can study the corresponding set of depths (Step 3 in QCD-F-OD procedure), which can be obtained by means of the function *outlier_detection()*.

```
>outlier_detection(FinancialData$data)$Depths
46          42          38          20          10          2           5           8
0.2555479   0.2584443   0.2627515   0.2643318   0.2643733   0.2650152   0.2659087   0.2662398
29          17          25          34          45          35          30          6
0.2713271   0.2734367   0.2736475   0.2759959   0.2760933   0.2766480   0.2766946   0.2772307
44          15          49          4           22          16          24          23
0.2779556   0.2791861   0.2793979   0.2801033   0.2816751   0.2817793   0.2819326   0.2827261
19          39          27          47          7           13          21          41
0.2832156   0.2833994   0.2834155   0.2840379   0.2844656   0.2845571   0.2856555   0.2864312
12          32          3           43          14          18          11          33
0.2876821   0.2878532   0.2891768   0.2898620   0.2902024   0.2914533   0.2918115   0.2922976
28          26          37          1           48          36          40          31
0.2923891   0.2924162   0.2928963   0.2931928   0.2968801   0.2974348   0.2979954   0.2981623
9           50
0.3004412   0.3025542
```

Fig. 7 displays a boxplot of the depth values. Note that only one point appears at the bottom of the graph. This point, which corresponds to the company Tesla, has an extreme (low) value according to the depth distribution. Therefore, Tesla could be regarded as the only anomalous company in dataset FinancialData. Note that this type of empirical methodology automatically determines the number of outliers. Similar analyses can be carried out when a dissimilarity matrix-based approach for outlier detection is considered.

4.5. Reducing the dimensionality of MTS objects with mlmts

Dimensionality reduction of MTS can be carried out by using some distance functions available in **mlmts**. The goal is to construct lower dimensional MTS objects preserving the information contained in the original series with high accuracy. Some approaches focus on removing redundant components. For instance, the procedure proposed in [53] (see Section 2.4) uses maximum values of the cross-correlations between components to decide which variables must be eliminated. This algorithm is implemented in **mlmts** by means of the function *dis_mcc()* as long as we set *features = TRUE*. We use this function to reduce the dimension of all the series in dataset RacketSports.

```
>reduced_dataset_mcc <- dis_mcc(RacketSports
  $data, max_lag = 10, delta = 0.7, features = T)
```

A maximum number of lags of $L = 10$ and a threshold value $\delta = 0.7$ were considered. Note that, the greater the value of δ, the lower the expected number of variables to be removed. Let's show the first MTS in the reduced dataset.

```
>head(reduced_dataset_mcc[[1]])
     [,1]        [,2]
[1,] 1.266676    0.031960
[2,] −2.180751   −1.725865
[3,] −0.943348   −1.523449
[4,] 0.440631    −1.086656
[5,] 1.562197    −0.183773
[6,] 0.248300    0.516694
```

The reduced series contains only the first and fifth components of the original MTS (all series in RacketSports have 6 dimensions). The remaining variables were eliminated because they show a substantial degree of correlation with the retained ones. Let's examine next the second series in the new dataset.

```
>head(reduced_dataset_mcc[[2]])
     [,1]        [,2]        [,3]        [,4]
[1,] 0.464568    0.281542    −0.237040   −0.170456
[2,] 7.017253    −1.483576   0.551318    −1.174547
[3,] 0.401548    −5.083115   0.588605    −2.397035
[4,] 0.401548    −5.083115   0.588605    −2.397035
[5,] −2.830047   −2.741396   0.463427    −2.269193
[6,] −6.471095   −1.829423   0.364882    −1.587370
```

In this case, four components were retained and only two removed, which means that the variables of the second series show a less degree of correlation than the ones of the first series. Note that, as shown in this example, the series in the new collection have generally different dimensions, since each original MTS is processed individually. For example, the components of the fifth MTS in RacketSports are barely correlated, thus resulting in a reduced series containing all the original variables.

```
>head(reduced_dataset_mcc[[5]])
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] −0.108107  0.799450  0.772173 −0.013317 −0.053267 −0.287644
[2,]  4.973278 −2.803670 −3.346607  1.592697  0.378199 −0.945497
[3,]  1.181158 −3.093719 −6.995264  1.438221 −0.034624 −0.926854
[4,] −1.893015 −1.686749 −4.629456 −0.799012 −0.154476 −0.127842
[5,] −13.361450 −7.628262 12.033613 −2.231906  1.917628  1.448875
[6,] −4.248398 10.361725  3.936902  0.375536  2.964334  1.262439
```

The parameter δ can be properly adjusted according to the number of variables one wishes to retain, thus giving the user the freedom to decide about the dimensionality of the reduced series.

Next, let's reduce the dimensionality of the first series in dataset RacketSports by employing an alternative procedure, namely, the technique PPCA, which divides a given MTS into different segments. The corresponding algorithm is implemented in **mlmts** through the function *dis_ppca()*.

```
>reduced_dataset_ppca <- dis_ppca(RacketSports
  $data, w = 3, features = T)

>head(reduced_dataset_ppca[[1]])
       [,1]        [,2]       [,3]
[1,] −0.2195899  3.78500495  1.8524667
[2,] −0.7901470  3.34970596 −1.7414870
[3,] −0.0527398  4.54178237 −0.8457695
[4,]  1.0397579  3.16751296  0.1407492
[5,]  1.1126066  0.01146494  1.5294547
[6,] −1.5327390 −2.72954395  1.7406453
```

The argument w was set to 3 in order to indicate that 3 segments are considered to obtain the average covariance matrix $\mathbf{\Sigma}_a$. Here, the reduced MTS has three variables, which means that the first three principal components are enough to capture the 90% of the variability (default). Clearly, different values of w could lead to very different results. In this regard, the user should examine the local patterns in the corresponding MTS to find a suitable value for w. Unlike the procedure based on maximal cross-correlations, the reduced series produced by PPCA do not contain a subset of the original variables, but a collection of synthetic variables creating via the standard PCA by taking into account the local information. Nevertheless, as with the standard PCA, one could inspect the corresponding eigenvectors to get insights into the influence of each one of the original variables in the principal components.

As a last remark, it is worth highlighting that the original collection of series could be replaced by the dataset produced by *dis_mcc()* or *dis_ppca()* before performing specific data mining tasks. In this way, one could expect a substantial improvement in computational efficiency, specially when the original series have a large number of variables.

4.6. Forecasting MTS with mlmts

Forecasting of multivariate series can be carried out in **mlmts** with the function *mts_forecasting()* implementing the FFMTS technique described in Section 2.5. Here, *mts_forecasting()* is used to obtain future values of a particular time series. Specifically, we consider the first MTS in dataset StandWalkJump and a value of 1 (default one) for the prediction horizon h and the maximum lag L. For evaluation purposes, we begin by splitting the corresponding time series into two periods, the training period, formed by all but the last observation, and the validation period, formed by the last observation.

```
>training_series <- StandWalkJump$data[[1]]
  [-2500,]
>validation_series <- StandWalkJump$data[[1]]
  [2500,]
```

The FFMTS procedure allows to fit an arbitrary regression model to the lag-embedding matrices. Let's start with standard linear regression (default) and compute the mean absolute error (MAE) associated with the corresponding model.

```
>predictions <- mts_forecasting(list
  (training_series))
>predictions_vector <- unlist(predictions)
>predictions_vector
[1] −0.04827054 0.03000827 −0.02603342 0.01958164
>mean(abs(predictions_vector-
  validation_series))
[1] 0.03117851
```

The MAE for the linear model is 0.031. However, this isolated value does not provide too much information. In fact, the considered error metric is frequently used to compare the predictive performance of several procedures. Let's compute the MAE for three alternative approaches, namely random forest, support vector machine, and a generalized additive model using splines. The regression model to be fitted to the lag-embedding matrices is specified in function *mts_forecasting()* by means of the argument *model_caret*.

```
>predictions_rf <- mts_forecasting(list
  (training_series), model_caret = 'rf')
>predictions_vector_rf <- unlist(predictions_rf)
>mean(abs(predictions_vector_rf-
  validation_series))
[1] 0.02228232

>predictions_svm <- mts_forecasting(list
  (training_series), model_caret = 'svmLinear')
>predictions_vector_svm <- unlist
  (predictions_svm)
>mean(abs(predictions_vector_svm-
  validation_series))
[1] 0.02397654

>predictions_gam <- mts_forecasting(list
  (training_series), model_caret = 'gamSpline')
>predictions_vector_gam <- unlist
  (predictions_gam)
>mean(abs(predictions_vector_gam-
  validation_series))
[1] 0.03117849
```

Table 5
MAE associated with four regression models when forecasting the first series in dataset StandWalkJump with method FFMTS. Different values of h and L are considered. The best results are shown in bold.

	$h=3$		$h=10$	
Model	$L=3$	$L=5$	$L=3$	$L=5$
Linear regression	**0.044**	0.036	0.084	0.099
Random forest	0.047	**0.028**	**0.040**	**0.024**
Support vector machine	0.047	0.039	0.074	0.150
Generalized linear model	**0.044**	0.036	0.084	0.099

The random forest and the support vector machine achieve a substantially lower prediction error than the linear model (0.022 and 0.024, respectively), while the generalized linear model attains a similar error than the latter. The predictive performance of any other regression model available in package **caret** can be examined in a similar way. Finally, let's analyze the behavior of the four considered methods for different values of the prediction horizon and the maximum lag. Specifically, we consider $h = 3, 10$ and $L = 3, 5$. The series is divided into training and validation periods in the same way as before. Table 5 contains the corresponding values of the MAE for each combination of the parameters.

The results in Table 5 indicate that the random forest yields the most accurate forecasts in most of the settings. Curiously, this method is able to improve its accuracy when increasing the prediction horizon, which usually means a more challenging forecasting task. In fact, all the remaining methods substantially decrease their performance when $h = 10$. In addition, the predictive ability of the random forest significantly improves when more lags are considered, which indicates that significant serial dependence exists beyond the third lag in the corresponding series. The remaining three procedures show a similar performance among themselves. Interestingly, the support vector machine behaves even worse than the linear model when $h = 10$ and $L = 5$, suggesting that this method is not suitable at all to forecast the considered MTS. In short, the results of this study suggest that the first series in dataset StandWalkJump follows a complex nonlinear structure which can be properly exploited by using a powerful model like the random forest to predict future values.

Note that the previous methodology can be replicated with any MTS we wish to forecast. In this way, package **mlmts** incorporates a flexible tool which allows the user not only to compute the corresponding point predictions, but also to determine the optimal forecasting model for a specific series.

5. Concluding remarks

The 21st century has witnessed a significant number of advances in the field machine learning of temporal data. Early works usually focused on UTS, but MTS have received a great deal of attention in the last decade. The R package **mlmts** is fundamentally an attempt to integrate different MTS data mining algorithms in a single framework, thus providing users with a common environment to check and compare the behavior of well-established methods. The main motivation behind the package is that, to the best of our knowledge, no previous R packages are available implementing several machine learning methods for MTS. Although the majority of functions in the package are associated with the computation of a distance matrix, the use of **mlmts** is not limited to clustering. In fact, visualization, supervised classification, outlier detection, dimensionality reduction and forecasting tasks can be also carried out in an effective and efficient manner. Several real and synthetic datasets which allow to illustrate the main methods implemented in the package are also included. Specifically, **mlmts** contains 28 of the 30 MTS collections provided in the well-known UEA multivariate time series classification archive [23,37], which are frequently used to test the performance of new methods. A reasonably complete description of the procedures available in **mlmts** is given in the first part of this paper to make clear the details and the scope behind the software. However, the readers particularly interested in some of the procedures are encouraged to check the corresponding key references, which are also provided in the manuscript. In the second part of the manuscript, the use of the package is illustrated by considering several examples where the implemented functions are executed with the UEA collections and the synthetic datasets. **mlmts** is under continuous development and we expect to perform frequent updates by incorporating future machine learning algorithms in the field of MTS data mining.

CRediT authorship contribution statement

Ángel López-Oriona: Conceptualization, Writing – review & editing, Methodology, Software, Visualization. **José A. Vilar:** Conceptualization, Supervision, Writing – review & editing, Project administration.

Data availability

No data was used for the research described in the article.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ángel López-Oriona reports financial support was provided by University of A Coruña. Ángel López-Oriona reports a relationship with University of A Coruña that includes: No has patent pending to no. none.

Acknowledgments

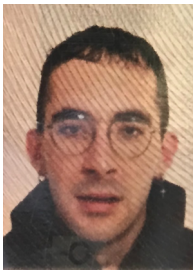
We are very grateful to professors Anthony Bagnall, Eamonn Keogh and coauthors for allowing us to share the datasets of the UEA archive with the R community and to the anonymous reviewers for their comments regarding the first version of this article.

This research has been supported by the Ministerio de Economía y Competitividad (MINECO) grants MTM2017-82724-R and PID2020-113578RB-100, the Xunta de Galicia (Grupos de Referencia Competitiva ED431C-2020-14), and the Centro de Investigación del Sistema Universitario de Galicia, "CITIC" grant ED431G 2019/01; all of them through the European Regional Development Fund (ERDF). This work has received funding for open access charge by University of A Coruña/CISUG.

References

[1] T.-C. Fu, A review on time series data mining, Eng. Appl. Artif. Intell. 24 (1) (2011) 164–181.
[2] A. Fakhrazari, H. Vakilzadian, A survey on time series data mining, in: 2017 IEEE International Conference on Electro Information Technology (EIT), IEEE, 2017, pp. 476–481.
[3] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, E. Keogh, Generalizing dtw to the multi-dimensional case requires an adaptive approach, Data Min. Knowl. Discov. 31 (1) (2017) 1–31.
[4] Y. Kakizawa, R.H. Shumway, M. Taniguchi, Discrimination and clustering for multivariate time series, J. Am. Stat. Assoc. 93 (441) (1998) 328–340.
[5] X. Wang, A. Wirth, L. Wang, Structure-based statistical features and multivariate time series clustering, in: Seventh IEEE International Conference on Data Mining (ICDM 2007), IEEE, 2007, pp. 351–360.
[6] P. D'Urso, E.A. Maharaj, Wavelets-based clustering of multivariate time series, Fuzzy Sets Syst. 193 (2012) 33–61.

[7] Á. López-Oriona, J.A. Vilar, Quantile cross-spectral density: A novel and effective tool for clustering multivariate time series, Expert Syst. Appl. 185 (2021) .
[8] E.A. Maharaj, Comparison and classification of stationary multivariate time series, Pattern Recogn. 32 (7) (1999) 1129–1138.
[9] K. Yang, C. Shahabi, A pca-based similarity measure for multivariate time series, in: Proceedings of the 2nd ACM international workshop on Multimedia databases, 2004, pp. 65–74.
[10] A. Singhal, D.E. Seborg, Clustering multivariate time-series data, J. Chemometrics 19 (8) (2005) 427–438.
[11] T.W. Liao, Clustering of time series data–a survey, Pattern Recogn. 38 (11) (2005) 1857–1874.
[12] S. Rani, G. Sikka, Recent techniques of clustering of time series data: a survey, Int. J. Comput. Appl. 52(15).
[13] Z. Bankó, J. Abonyi, Correlation based dynamic time warping of multivariate time series, Expert Syst. Appl. 39 (17) (2012) 12814–12823.
[14] J. Mei, M. Liu, Y.-F. Wang, H. Gao, Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification, IEEE Trans. Cybern. 46 (6) (2015) 1363–1374.
[15] T. Górecki, M. Łuczak, Multivariate time series classification with parametric derivative dynamic time warping, Expert Syst. Appl. 42 (5) (2015) 2305–2312.
[16] A. Zagorecki, A versatile approach to classification of multivariate time series data, in: Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015 5 (2015) 407–410.
[17] Á. López-Oriona, J.A. Vilar, F4: An all-purpose tool for multivariate time series classification, Mathematics 9 (23) (2021) 3051.
[18] C. Li, L. Khan, B. Prabhakaran, Real-time classification of variable length multi-attribute motions, Knowl. Inf. Syst. 10 (2) (2006) 163–183.
[19] C. Li, L. Khan, B. Prabhakaran, Feature selection for classification of variable length multiattribute motions, in: Multimedia data mining and knowledge discovery, Springer, 2007, pp. 116–137.
[20] X. Weng, J. Shen, Classification of multivariate time series using locality preserving projections, Knowl.-Based Syst. 21 (7) (2008) 581–587.
[21] P. Schäfer, U. Leser, Multivariate time series classification with weasel+ muse, arXiv preprint arXiv:1711.11343.
[22] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate lstm-fcns for time series classification, Neural Networks 116 (2019) 237–245.
[23] A. Bagnall, H.A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. Keogh, The uea multivariate time series classification archive, 2018, arXiv preprint arXiv:1811.00075.
[24] X. Weng, J. Shen, Detecting outlier samples in multivariate time series dataset, Knowl.-based Syst. 21 (8) (2008) 807–812.
[25] R.J. Hyndman, E. Wang, N. Laptev, Large-scale unusual time series detection, in: 2015 IEEE international conference on data mining workshop (ICDMW), IEEE, 2015, pp. 1616–1619.
[26] Á. López-Oriona, J.A. Vilar, Outlier detection for multivariate time series: A functional data approach, Knowl.-Based Syst. 233 (2021) .
[27] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2020, URL: https://www.R-project.org/.
[28] T. Giorgino, Computing and visualizing dynamic time warping alignments in R: The dtw package, J. Stat. Softw. 31 (7) (2009) 1–24.
[29] T. Giorgino, Computing and visualizing dynamic time warping alignments in r: the dtw package, J. Stat. Software 31 (2009) 1–24.
[30] A.M. Brandmaier, pdc: An R package for complexity-based clustering of time series, J. Stat. Softw. 67 (5) (2015) 1–23.
[31] P. Montero, J.A. Vilar, TSclust: Time series clustering utilities, R package version 1.2.1 (2014). URL: http://CRAN.R-project.org/package=TSclust.
[32] U. Mori, A. Mendiburu, J.A. Lozano, Distance measures for time series in R: The TSdist package, R J. 8 (2) (2016) 451–459.
[33] A. López-Oriona, J. A. Vilar, mlmts: Machine Learning Algorithms for Multivariate Time Series, r package version 1.0.1 (2022). URL: https://CRAN.R-project.org/package=mlmts.
[34] M.M. Fréchet, Sur quelques points du calcul fonctionnel, Rendiconti del Circolo Matematico di Palermo (1884–1940) 22 (1) (1906) 1–72.
[35] P. Montero, J.A. Vilar, Tsclust: An r package for time series clustering, J. Stat. Softw. 62 (2015) 1–43.
[36] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, KDD workshop, Vol. 10, Seattle, WA, USA, 1994, pp. 359–370.
[37] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Disc. 35 (2) (2021) 401–449.
[38] P. D'Urso, L. De Giovanni, R. Massari, Trimmed fuzzy clustering of financial time series based on dynamic time warping, Ann. Oper. Res. 299 (1) (2021) 1379–1395.
[39] A. Singhal, D.E. Seborg, Pattern matching in multivariate time series databases using a moving-window approach, Ind. Eng. Chem. Res. 41 (16) (2002) 3822–3838.
[40] E.G.S. Nascimento, O. de Lira Tavares, A.F. De Souza, A cluster-based algorithm for anomaly detection in time series using mahalanobis distance, in: Proceedings on the International Conference on Artificial Intelligence (ICAI), The Steering Committee of The World Congress in Computer Science, Computer, 2015, p. 622.
[41] P. D'Urso, E.A. Maharaj, Autocorrelation-based fuzzy clustering of time series, Fuzzy Sets Syst. 160 (24) (2009) 3565–3589.
[42] A.M. Alonso, D. Peña, Clustering time series by linear dependency, Stat. Comput. 29 (4) (2019) 655–676.
[43] Hormann, Kidzinski, freqdom: Frequency Domain Based Analysis: Dynamic PCA, r package version 2.0.1 (2017). URL: https://CRAN.R-project.org/package=freqdom.
[44] B. Whitcher, waveslim: Basic Wavelet Routines for One-, Two-, and Three-Dimensional Signal Processing, r package version 1.8.2 (2020). URL: https://CRAN.R-project.org/package=waveslim.
[45] B. Lafuente-Rego, J.A. Vilar, Clustering of time series using quantile autocovariances, Adv. Data Anal. Classification 10 (3) (2016) 391–415.
[46] J. Baruník, T. Kley, Quantile coherency: A general measure for dependence between cyclical economic variables, Econometrics J. 22 (2) (2019) 131–152.
[47] T. Kley, Quantile-based spectral analysis in an object-oriented framework and a reference implementation in R: The quantspec package, J. Stat. Softw. 70 (3) (2016) 1–27, https://doi.org/10.18637/jss.v070.i03.
[48] D. Piccolo, A distance measure for classifying arima models, J. Time Ser. Anal. 11 (2) (1990) 153–164.
[49] D. Melo, G. Garcia, A. Hubbe, A.P. Assis, G. Marroig, Evolqg-an r package for evolutionary quantitative genetics, F1000Research 4.
[50] X. Weng, J. Shen, Classification of multivariate time series using two-dimensional singular value decomposition, Knowl.-Based Syst. 21 (7) (2008) 535–539.
[51] X. Wan, H. Li, L. Zhang, Y.J. Wu, Dimensionality reduction for multivariate time-series data mining, J. Supercomput. 78 (7) (2022) 9862–9878.
[52] H. He, Y. Tan, Unsupervised classification of multivariate time series using vpca and fuzzy clustering with spatial weighted matrix distance, IEEE Trans. Cybern. 50 (3) (2018) 1096–1105.
[53] A. Egri, I. Horváth, F. Kovács, R. Molontay, K. Varga, Cross-correlation based clustering and dimension reduction of multivariate time series, in: 2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES), IEEE, 2017, pp. 000241–000246.
[54] R. Fraiman, G. Muniz, Trimmed means for functional data, Test 10 (2) (2001) 419–440.
[55] H. Li, Multivariate time series clustering based on common principal component analysis, Neurocomputing 349 (2019) 239–247.
[56] P. Montero-Manso, R.J. Hyndman, Principles and algorithms for forecasting groups of time series: Locality and globality, Int. J. Forecast. 37 (4) (2021) 1632–1653.
[57] M. Kuhn, caret: Classification and Regression Training, r package version 6.0-91 (2022). URL: https://CRAN.R-project.org/package=caret.
[58] G. Dudnik, mltest: Classification Evaluation Metrics, r package version 1.0.1 (2018). URL: https://CRAN.R-project.org/package=mltest.
[59] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, cluster: Cluster Analysis Basics and Extensions, r package version 2.1.2 — For new features, see the 'Changelog' file (in the package source) (2021). URL: https://CRAN.R-project.org/package=cluster.
[60] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1) (1985) 193–218.
[61] L. Mouselimis, ClusterR: Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans, K-Medoids and Affinity Propagation Clustering, r package version 1.2.6 (2022). URL: https://CRAN.R-project.org/package=ClusterR.



Ángel López-Oriona is a researcher at the University of A Coruña, Spain. He received his bachelor's degree in Mathematics and his master's degree in Statistics both from the University of Santiago de Compostela, Spain, and his masters's degree in Big Data from the European University of Madrid, Spain. He authored some papers in JCR journals and participated in several international conferences. His research interests include computational statistics and behavioral economics. He loves travelling to exotic places with monkeys, jungles and high-quality beer. His favorite cities are Budapest, Kyoto and Sydney. He spends his Monday mornings having breakfast at the closest McDonald's while reading his Kindle.

José Antonio Vilar is a full professor at the University of A Coruña, Spain. He received his bachelor's degree in Mathematics and his Ph.D. in Statistics both from the University of Santiago de Compostela, Spain. He participated in more than 30 competitive projects and authored more than 40 papers in JCR journals. His past research focused on nonparametric inference. His recent research interests include clustering and classification of time series.