Journal of Computational Science 76 (2024) 102233

Contents lists available at ScienceDirect

Journal of Computational Science

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

Analyzing categorical time series with the R package **ctsfeatures**

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

[a] King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia
[b] Research Group MODES, Research Center for Information and Communication Technologies (CITIC), University of A Coruña, 15071 A Coruña, Spain

ARTICLE INFO

Keywords:
ctsfeatures
R package
Categorical time series
Feature extraction
Association measures

ABSTRACT

Time series data are ubiquitous nowadays. Whereas most of the literature on the topic deals with real-valued time series, categorical time series have received much less attention. However, the development of data mining techniques for this kind of data has substantially increased in recent years. The R package **ctsfeatures** offers users a set of useful tools for analyzing categorical time series. In particular, several functions allowing the extraction of well-known statistical features and the construction of illustrative graphs describing underlying temporal patterns are provided in the package. The output of some functions can be employed to perform traditional machine learning tasks including clustering, classification and outlier detection. The package also includes two datasets of biological sequences introduced in the literature for clustering purposes, one dataset of sleep stages, and three interesting synthetic databases. In this work, 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 valuable tools provided by **ctsfeatures**.

1. Introduction

Traditionally, most of the literature on time series analysis has focused on real-valued time series, while the study of time series with discrete response has received much less attention even though they arise in a natural way in many applications. Examples include weekly counts of new infections with a specific disease [1], temporal records of EEG sleep states [2], the stochastic modeling of DNA sequence data [3,4] and financial time series [5], and the use of hidden Markov processes to deal with protein sequences [6], among others. An excellent introduction to the topic of discrete-valued time series is provided by [7], including classical models, recent advances, relevant references and specific application areas.

Categorical time series (CTS) are characterized by taking values on a qualitative range consisting of a finite number of categories, which is referred to as ordinal range, if the categories exhibit a natural ordering, or nominal range, otherwise. In this work, the most general case of nominal range is considered. Indeed, dealing with unordered qualitative outcomes implies that some classical analytic tools are no longer useful, since categories cannot be ranked and no algebraic operations can be performed on them. This way, standard measures of location (mean, median, quantiles), dispersion (standard deviation, range) and serial dependence (autocorrelation, partial autocorrelation) are not defined in the nominal setting, and alternative measures considering the qualitative nature of the range are needed for the analysis of CTS. Although the case of ordinal time series falls outside the scope of this article, it is worth mentioning that the statistical features described throughout this work are also useful with ordinal responses. However, in this case, additional features (measuring dispersion, skewness, serial dependence,...) can be used to take advantage of the latent ordering existing in the series' range (see e.g., [8]). The importance of considering the ordinal nature of the series is properly discussed and illustrated in the recent work by the authors [9], where the ordinal series are characterized by cumulative probabilities describing both the marginal properties and the serial dependence patterns.

While early articles about CTS mainly focused on problems of statistical inference and modeling, several works addressing the construction of machine learning algorithms for categorical series have been published in recent years. For instance, in the context of clustering, [10] proposed an algorithm based on a mixture of first order Markov models to group time series recording navigation patterns of users of a web site. A different strategy was considered in [11], where a dissimilarity measure combining both closeness of raw categorical values and proximity between dynamic behaviors is used as input to a modified version of the K-modes algorithm. Alternatively, a well-known approach in time series analysis consists of representing the series by means of extracted features capturing their main properties (see e.g., [12] and specific software packages considering

* Corresponding author at: Research Group MODES, Research Center for Information and Communication Technologies (CITIC), University of A Coruña, 15071 A Coruña, Spain.
E-mail addresses: a.oriona@udc.es, angel.lopezoriona@kaust.edu.sa (Á. López-Oriona), jose.vilarf@udc.es (J.A. Vilar).

https://doi.org/10.1016/j.jocs.2024.102233
Received 27 March 2023; Received in revised form 29 January 2024; Accepted 6 February 2024
Available online 13 February 2024

this via such as [13,14]). Dealing with CTS, a feature-based clustering algorithm aimed at detecting groups of series with similar dependence structures was introduced by [15]. Their approach consists of replacing the original series by a set of informative statistics extracted from each one of them, which are then used as input to a standard clustering procedure. The feature-based approach has also been employed for other tasks as CTS classification. For example, [16] proposed a novel approach relying on the so-called spectral envelope and its corresponding set of optimal scalings, which characterizes oscillatory patterns in a categorical series. Besides clustering and classification, there are other interesting and challenging problems related to categorical sequences, as anomaly detection [17]. Previous references highlight the remarkable growth that CTS mining has recently undergone.

According to these considerations, it seems clear that the development of software specifically designed to deal with CTS is of great importance. However, although there exist several packages revolving around the analysis of categorical data without a temporal nature (e.g., the R [18] package **confreq** [19,20] or the `Python` [21] package **categorical-encoding**), only a few computing tools are focused on categorical sequences. Moreover, these tools are often restricted to one specific task, application domain or class of categorical models. For instance, the R package **TraMineR** [22] implements a collection of dissimilarity measures between CTS (see [23]) related to the so-called sequence analysis [24], a topic of great interest in social sciences. Most of these metrics attempt to detect differences concerning the order in which successive categories appear, the timing and the duration of the spells for different categories. Another interesting tool devoted to sequence analysis is the `STATA` [25] package **SADI** [26], which incorporates graphical summaries and tools for cluster analysis besides computing several dissimilarity criteria. Some other packages are designed to deal with a particular type of model for CTS. For example, the R package **HMM** [27] provides several functions to perform statistical inference for Hidden Markov Models (HMM). Note that, in terms of exploratory analyses, focusing on a specific class of models is too restrictive, since most real CTS datasets are expected to contain series generated from different types of stochastic processes. Although their usefulness is beyond doubt, none of the available software packages for handling CTS provides the necessary tools to compute classical statistical features for categorical series.

The goal of this article is to present the R package **ctsfeatures**, which contains several functions to compute well-known statistical features for CTS. Besides providing a valuable description of the structure of the series, these features can be used as input for traditional machine learning procedures, as clustering, classification and outlier detection algorithms. In addition, **ctsfeatures** includes some visualization tools, as well as serial dependence plots, which can be seen as an extension of the autocorrelation plots for numerical time series. The two databases of biological sequences introduced in [15] are also available in the package, along with other real database with sequences of sleep stages used in [16] and several synthetic datasets formed by CTS generated from different stochastic processes. These data collections allow the users to test the functions provided in the package.

In summary, **ctsfeatures** intends to integrate a set of simple but powerful functions for the statistical analysis of CTS into a single framework. Implementation was carried out by using the open-source R programming language due to the role of R as the most used programming language for statistical computing. The package **ctsfeatures** is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=ctsfeatures.

The rest of the paper is structured as follows. A summary of relevant features to analyze marginal properties and serial dependence of categorical time series is presented in Section 2. Furthermore, features measuring cross-dependence between categorical and numerical processes are also introduced. The main functions implemented in **ctsfeatures** and the available datasets are listed and discussed in Section 3. Section 4 is devoted to illustrate the functionality of the package through several examples considering the synthetic data and the biological databases included in **ctsfeatures**. In addition, the way in which the outputs of some functions can be used to perform classical machine learning tasks is properly described. Lastly, some concluding remarks are provided in Section 5.

2. Analyzing marginal and serial properties of categorical time series

Let $\{X_t, t \in \mathbb{Z}\}$ be a one-dimensional categorical stochastic process taking values on a finite number r of unordered qualitative categories, which are coded with the integer numbers from 1 to r. The range of the process is denoted by $\mathcal{V} = \{1, \ldots, r\}$. In order to ensure that the statistical features implemented in **ctsfeatures** to characterize the marginal properties and the serial dependence of X_t are well-defined, we will assume that X_t is a weakly stationary process. Notice that the usual notion of weak stationarity for real-valued series refers to the invariance in t of the expectations $E(X_t)$ and the covariances $Cov(X_t, X_{t+l})$, which are not defined for categorical data. Thus, among the different approaches proposed to define weak stationary for CTS, we assume that X_t satisfies the concept of *bivariate stationarity* introduced in [4]. According to Definition and Theorem 3.1 in [4], a categorical time series X_t satisfies the bivariate stationarity property if the pairwise joint distribution of (X_t, X_{t+l}) is invariant in t for each lag $l \in \mathbb{Z}$.

It is often useful to represent $\{X_t, t \in \mathbb{Z}\}$ as the binary r-variate process $\{\boldsymbol{Y}_t = (Y_{t,1}, \ldots, Y_{t,r})^\top, t \in \mathbb{Z}\}$, where $Y_{t,j} = 1$ iff $X_t = j$ and 0 otherwise, for $j = 1, \ldots, r$. In other words, $X_t = j$ iff $\boldsymbol{Y}_t = \boldsymbol{e}_j$, with $\boldsymbol{e}_j$ being the unit vector of length r taking the value 1 in the jth position and 0 in all others. This representation, usually referred to as *binarization* of $\{X_t, t \in \mathbb{Z}\}$, makes it easier to employ algebraic expressions to introduce some concepts and is frequently used in this context [7]. As a simple example, using the binarization, the vector of frequencies of the different categories in a realization of X_t can be obtained as the sum of the corresponding binarized observations.

According to Theorem 3.1 in [4], bivariate stationarity implies *marginal stationarity*, which states that all the marginal probabilities are invariant in t. Hereafter, the marginal distribution of X_t is denoted by $\boldsymbol{p} = (p_1, \ldots, p_r)^\top$, with $p_j = P(X_t = j) = P(\boldsymbol{Y}_t = \boldsymbol{e}_j)$, $j = 1, \ldots, r$.

In order to describe the marginal properties of the process, features measuring dispersion could be considered. However, categorical dispersion is not a simple concept due to the fact that the range of $\{X_t, t \in \mathbb{Z}\}$ is intrinsically qualitative and the expectation of the process cannot be computed in the standard way. An intuitive approach to understand the degree of dispersion of an arbitrary variable is in terms of the capability to predict its outcome. That is, the smaller the dispersion, the better we can predict the result. This intuitive understanding of dispersion can be adapted for categorical variables by simply considering the marginal probabilities: maximal dispersion when all probabilities p_j are equal to each other (uniform marginal distribution), and minimal dispersion if $p_i = 1$ for some $i \in \mathcal{V}$, and $p_j = 0$ for $j \neq i$ (one-point distribution). Then, any concentration index mapping the extremes of its range to their values for the one-point and uniform distributions can be used as measure of dispersion. Several indexes have been proposed in the literature (see a brief review in Appendix A of [4] and the references therein). Three of these measures are the Gini index, the entropy and the Chebycheff dispersion, which are given in the first rows of Table 1. All of them have range $[0, 1]$, reaching the value 0 for the one-point distribution (minimal dispersion) and the value of 1 for the uniform distribution (maximal dispersion). The asymptotic properties of the sample versions of these measures in the i.i.d. case can be seen in [7].

Table 1
Some relevant features to measure dispersion and serial dependence of a one-dimensional categorical stochastic process (top) and cross-dependence between the categorical and numerical components of a two-dimensional mixed process (bottom).

One-dimensional categorical processes Measure	Definition	Range	Type
Gini index	$g = \frac{r}{r-1}\left(1 - \sum_{i=1}^{r} p_i^2\right)$	$[0, 1]$	Marginal
Entropy	$e = \frac{-1}{\ln(r)} \sum_{i=1}^{r} p_i \ln p_i$	$[0, 1]$	Marginal
Chebycheff dispersion	$c = \frac{m}{m-1}(1 - \max_i p_i)$	$[0, 1]$	Marginal
Goodman and Kruskal's τ	$\tau(l) = \frac{\sum_{i,j=1}^{r} \frac{p_{ij}(l)^2}{p_j} - \sum_{i=1}^{r} p_i^2}{1 - \sum_{i=1}^{r} p_i^2}$	$[0, 1]$	Serial
Goodman and Kruskal's λ	$\lambda(l) = \frac{\sum_{j=1}^{r} \max_i p_{ij}(l) - \max_i p_i}{1 - \max_i p_i}$	$[0, 1]$	Serial
Uncertainty coefficient	$u(l) = -\frac{\sum_{i,j=1}^{r} p_{ij}(l) \ln\left(\frac{p_{ij}(l)}{p_i p_j}\right)}{\sum_{i=1}^{r} p_i \ln p_i}$	$[0, 1]$	Serial
Pearson measure	$X_n^2(l) = n \sum_{i,j=1}^{r} \frac{(p_{ij}(l) - p_i p_j)^2}{p_i p_j}$	$[0, n(r-1)]$	Serial
Φ^2-measure	$\Phi^2(l) = X_n^2(l)/n$	$[0, r-1]$	Serial
Sakoda measure	$p^*(l) = \sqrt{\frac{r\Phi^2(l)}{(r-1)(1+\Phi^2(l))}}$	$[0, 1]$	Serial
Cramer's v	$v(l) = \Phi(l)/\sqrt{r-1}$	$[0, 1]$	Serial
Cohen's κ	$\kappa(l) = \frac{\sum_{j=1}^{r}(p_{jj}(l) - p_j^2)}{1 - \sum_{i=1}^{r} p_i^2}$	$\left[-\frac{\sum_{i=1}^{r} p_i^2}{1 - \sum_{i=1}^{r} p_i^2}, 1\right]$	Serial
Total correlation	$\Psi(l) = \frac{1}{r^2} \sum_{i,j=1}^{r} \psi_{ij}(l)^2$	$[0, 1]$	Serial
Two-dimensional processes with numerical and categorical components **Measure**	**Definition**	**Range**	**Type**
Total mixed cross-correlation	$\Psi_1^*(l) = \frac{1}{r} \sum_{i=1}^{r} \psi_i^*(l)^2$	$[0, 1]$	Serial
Total mixed q-cross-correlation	$\Psi_2^*(l) = \frac{1}{r} \sum_{i=1}^{r} \int_0^1 \psi_i^\rho(l)^2 d\rho$	$[0, 1]$	Serial

To analyze the serial dependence structure of $\{X_t, t \in \mathbb{Z}\}$ for a given lag $l \in \mathbb{Z}$, it is interesting to consider the lagged joint probabilities $p_{ij}(l)$ and the lagged conditional probabilities $p_{i|j}(l)$, for $i, j = 1, \ldots, r$, which are defined by

$$\begin{aligned} p_{ij}(l) &= P(X_t = i, X_{t-l} = j) = P(\boldsymbol{Y}_t = \boldsymbol{e}_i, \boldsymbol{Y}_{t-l} = \boldsymbol{e}_j), \\ p_{i|j}(l) &= P(X_t = i | X_{t-l} = j) = P(\boldsymbol{Y}_t = \boldsymbol{e}_i | \boldsymbol{Y}_{t-l} = \boldsymbol{e}_j) = \frac{p_{ij}(l)}{p_j}. \end{aligned} \tag{1}$$

Based on probabilities in Eq. (1), [4] introduce notions of perfect serial independence and dependence for a categorical process. Specifically, we have perfect serial independence at lag $l \in \mathbb{Z}$ if and only if $p_{ij}(l) = p_i p_j$ for any $i, j \in \mathcal{V}$. On the other hand, we have perfect serial dependence at lag $l \in \mathbb{Z}$ if and only if the conditional distribution $p_{\cdot|j}(l)$ is a one-point distribution for any $j \in \mathcal{V}$. This way, knowledge about X_{t-l} does not help at all in predicting the value of X_t in a perfect serially independent process, while X_t is completely determined from X_{t-l} under perfect serial dependence. Note that these concepts describe the so-called unsigned dependence of the process X_t. However, it is often desirable to know whether a process tends to stay in the state it has reached (positive dependence) or, on the contrary, the repetition of the same state after l steps is infrequent (negative dependence). In other terms, by analogy with the autocorrelation function of a numerical process, which takes positive or negative values, it is interesting to introduce a signed dependence measure for categorical processes. Again following [4], provided that perfect serial dependence holds, we have perfect positive (negative) serial dependence if $p_{i|i}(l) = 1$ ($p_{i|i}(l) = 0$) for all $i \in \mathcal{V}$.

Once the notions of independence, perfect dependence and signed dependence at lag l have been introduced, it is interesting to have available specific measures accounting for the amount and type of existing dependence. For this purpose, some association measures have been proposed in the literature. As a general rule, a measure of association between two nominal variables evaluates the level of mutual relationship between both of them, and it may be used to account for the amount of dependence if the limits of its range are related to the extreme cases of independence and perfect dependence. For instance, a desirable property to relate association and dependence is that the association measure achieves the minimum value of its range when both variables are independent.

Some of the most important association measures available in the literature to describe the serial dependence structure of a categorical process at lag l are listed in the upper part of Table 1. Note that all of them are based on the probabilities introduced in Eqs. (1) and (3), although they can be organized at different categories according to how they are constructed.

The measures $\tau(l)$ and $\lambda(l)$ of Goodman and Kruskal and the uncertainty coefficient $u(l)$ form a first group of measures of unsigned association. All of them are aimed at assessing the proportional reduction of any dispersion measure of X_t induced by X_{t-l}. In fact, given a specific measure of dispersion $H(X_t)$ (such as g, e and c in Table 1), one can define the conditional dispersion $H(X_t|X_{t-l} = j)$ replacing the marginal probabilities

p_i by the lagged conditional probabilities $p_{i|j}(l)$ in the definition of H. Then, $\mathbb{E}(H(X_t|X_{t-l})) = \sum_{j=1}^{r} H(X_t|X_{t-l}=j)\,p_j$ and a measure of association between X_t and X_{t-l} can be introduced as

$$A_H(l) = \frac{H(X_t) - \mathbb{E}(H(X_t|X_{t-l}))}{H(X_t)} = 1 - \frac{\mathbb{E}(H(X_t|X_{t-l}))}{H(X_t)}. \tag{2}$$

Thus, $A_H(l)$ provides information about the proportional part of the variability of X_t accounted for X_{t-l}. The Goodman and Kruskal's τ, the Goodman and Kruskal's λ and the uncertainty coefficient are association measures in the form $A_H(l)$ in Eq. (2), which result from selecting as dispersion measure, $H(\cdot)$, the Gini index (g), the Chebycheff dispersion (c), and the entropy (e), respectively. The three measures satisfy suitable properties to assess unsigned dependence. Under independence, these measures take the value zero, although the inverse implication only holds for the Goodman and Kruskal's τ and the uncertainty coefficient. In the three cases, we have perfect dependence iff they take the value one. It is worth remarking that these measures are not symmetric.

A second group in Table 1 is formed by measures of unsigned association derived from the Pearson's χ^2-statistic. Specifically, the Pearson measure, $X_n^2(l)$, computes the squared differences between the joint probabilities $p_{ij}(l)$ and their corresponding factorization under independence, $p_i p_j$. Similarly, the Φ^2-measure, the Sakoda measure, and the Cramer's v are based on the Pearson's X_n^2. In all cases, independence occurs iff the corresponding measure takes the value zero, while perfect dependence is present iff the maximum value in the range is achieved (which differs from one for the Pearson and Φ^2 measures). Unlike the measures in the first group, these four measures are symmetric.

While previous measures do not shed light on the signed dependence patterns, the Cohen's $\kappa(l)$ can take either positive or negative values, since the differences between probabilities in the numerator are not squared. In fact, $\kappa(l)$ takes its lowest (highest) possible value iff X_t and X_{t-l} exhibit perfect negative (positive) dependence. Note that $\kappa(l)$ is symmetric.

All of the quantities in the upper part of Table 1 have been extensively used in the CTS literature. Several examples related to the interpretation of these measures are provided in [7], while their asymptotic properties for a specific type of categorical models are given in [28]. It is also worth remarking that the assumption of bivariate stationarity implies the invariance in t of these association measures, thus ensuring that all of them are well-defined.

An alternative way of evaluating serial dependence within a categorical process is by considering pairwise correlations in the binarization of X_t [15], that is, obtaining

$$\psi_{ij}(l) = Corr(Y_{t,i}, Y_{t-l,j}) = \frac{p_{ij}(l) - p_i p_j}{\sqrt{p_i(1-p_i)p_j(1-p_j)}}. \tag{3}$$

By construction, features in Eq. (3) play a similar role as the autocorrelation function of a numerical stochastic process, which accounts for their capability to characterize the underlying dependence. In fact, a clustering algorithm based on the values $\psi_{ij}(l)$ was proposed by [15] and exhibited a high accuracy grouping together series with similar underlying dependence patterns.

In practice, the quantities in the upper part of Table 1 must be estimated from a T-length realization of the process, $\overline{X}_t = \{\overline{X}_1, \ldots, \overline{X}_T\}$. For each measure, a standard estimate is directly obtained by replacing in the corresponding expression the probabilities p_i and $p_{ij}(l)$ by their respective unbiased estimates $\widehat{p}_i$ and $\widehat{p}_{ij}(l)$ given by

$$\begin{aligned} \widehat{p}_i &= \frac{N_i}{T}, \text{ with } N_i = \sum_{t=1}^{T} \mathbb{1}\left(\overline{X}_t = i\right), \\ \widehat{p}_{ij}(l) &= \frac{N_{ij}(l)}{T-l}, \text{ with } N_{ij}(l) = \sum_{t=l+1}^{T} \mathbb{1}\left((\overline{X}_t, \overline{X}_{t-l}) = (i,j)\right), \end{aligned} \tag{4}$$

for $i, j = 1, \ldots, r$, and where $\mathbb{1}(\mathcal{A})$ stands for the indicator function taking the value 1 if $\mathcal{A}$ happens and 0 otherwise. In all cases, the symbol $\widehat{\cdot}$ is used to denote the corresponding estimates (e.g., $\widehat{g}$ refers to the estimate of the Gini index). These estimates are often used to construct statistical procedures aimed at testing the serial independence of a given categorical process at a given lag l (see Section 3.2).

Another interesting issue when dealing with a categorical process $\{X_t, t \in \mathbb{Z}\}$ is to measure its degree of cross-dependence with a given real-valued process. Let $\{Z_t, t \in \mathbb{Z}\}$ be a strictly stationary real-valued process with variance σ^2. Then, for $i = 1, \ldots, r$, the level of linear dependence between the ith category of X_t and the process Z_t at a given lag $l \in \mathbb{Z}$ can be evaluated by means of

$$\psi_i^*(l) = Corr(Y_{t,i}, Z_{t-l}) = \frac{Cov(Y_{t,i}, Z_{t-l})}{\sqrt{p_i(1-p_i)\sigma^2}}, \tag{5}$$

where $Y_{t,i}$ is the ith component of the binarization of X_t.

Cross-correlations $\psi_i^*(l)$ are based on expected values and constitute the traditional way to examine linear dependence. However, uncorrelated variables might exhibit dependence in different parts of the joint distribution. A deeper analysis of cross-dependence can be performed by examining dependence in quantiles. More specifically, by using the quantities

$$\psi_i^{\rho}(l) = Corr\Big(Y_{t,i}, \mathbb{1}\big(Z_{t-l} \le q_{Z_t}(\rho)\big)\Big) = \frac{Cov\Big(Y_{t,i}, \mathbb{1}\big(Z_{t-l} \le q_{Z_t}(\rho)\big)\Big)}{\sqrt{p_i(1-p_i)\rho(1-\rho)}}, \tag{6}$$

for $i = 1, \ldots, r$, where $\rho \in (0,1)$ is an arbitrary probability level and $q_{Z_t}(\cdot)$ denotes the quantile function of the process Z_t. By considering different values for ρ, $\psi_i^{\rho}(l)$ provide measures of dependence at lag l between the ith category of X_t and the probabilities to exceed a range of quantiles of Z_t, which leads to a more comprehensive picture of the underlying dependence and allows us to detect different types of dependence structures.

In order to have available a global measure of linear cross-correlation between categorical and numerical processes, the features $\psi_i^*(l)$ can be used to define the *total mixed cross-correlation at lag l* given by

$$\Psi_1^*(l) = \frac{1}{r}\sum_{i=1}^{r} \psi_i^*(l)^2. \tag{7}$$

Analogously, a measure of the quantile cross-correlation between both processes, herein referred to as *total mixed q-cross-correlation at lag l*, can be defined as

$$\Psi_2^*(l) = \frac{1}{r}\sum_{i=1}^{r}\int_0^1 \psi_i^{\rho}(l)^2 d\rho. \tag{8}$$

By definition, both measures $\Psi_1^*(l)$ and $\Psi_2^*(l)$ (see the lower part of Table 1) take values on $[0,1]$, reaching the value 0 in the case of null cross-dependence between X_t and Z_t, and larger values when a stronger degree of cross-dependence between both processes is present.

In practice, given a T-length realization $(\overline{X}_t, \overline{Z}_t) = \{(\overline{X}_1, \overline{Z}_1), \ldots, (\overline{X}_T, \overline{Z}_T)\}$ of the two-dimensional process $\{(X_t, Z_t), t \in \mathbb{Z}\}$, respective estimates of $\psi_i^*(l)$ and $\psi_i^{\rho}(l)$ can be constructed by considering

$$\begin{aligned}
\widehat{\psi}_i^*(l) &= \frac{\widehat{Cov}(Y_{t,i}, Z_{t-l})}{\sqrt{\widehat{p}_i(1-\widehat{p}_i)\widehat{\sigma}^2}},\\
\widehat{\psi}_i^{\rho}(l) &= \frac{\widehat{Cov}\Big(Y_{t,i}, \mathbb{1}\big(Z_{t-l} \le \widehat{q}_{Z_t}(\rho)\big)\Big)}{\sqrt{\widehat{p}_i(1-\widehat{p}_i)\rho(1-\rho)}},
\end{aligned} \tag{9}$$

where $\widehat{Cov}(\cdot,\cdot)$ denotes the standard estimate of the covariance between two random variables, and $\widehat{\sigma}^2$ and $\widehat{q}_{Z_t}(\cdot)$ are standard estimates of the variance and the quantile function of process Z_t computed from the realization $\overline{Z}_t$. Then, estimates in Eq. (9) allow to estimate the global measures $\Psi_1^*(l)$ and $\Psi_2^*(l)$ by computing $\widehat{\Psi}_1^*(l) = \frac{1}{r}\sum_{i=1}^{r}\widehat{\psi}_i^*(l)^2$ and $\widehat{\Psi}_2^*(l) = \frac{1}{r}\sum_{i=1}^{r}\int_0^1 \widehat{\psi}_i^{\rho}(l)^2 d\rho$, respectively.

So far, the features introduced to characterize dependence are defined in the time domain, but valuable features in the frequency domain have also been proposed in the literature. A well-known spectral feature is the so-called *spectral envelope* [2,29]. The idea is to assign numerical values (*scaling*) to each of the categories of the process and then perform a spectral analysis of the resulting discrete-valued time series. Each scaling is properly selected to emphasize a specific cyclic pattern existing in the categorical process. Specifically, for $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_r)^\top \in \mathbb{R}^r$, $X_t(\boldsymbol{\gamma}) = \boldsymbol{\gamma}^\top \boldsymbol{Y}_t$ denotes the real-valued stationary time series corresponding to the scaling that assigns the ith category of X_t the value γ_i, $i = 1, \ldots, r$. For a given frequency ω, the vector $\boldsymbol{\gamma} = \boldsymbol{\gamma}(\omega)$ is determined by following the approach in [2,29]. First, the Fourier transform is applied to compute the spectral density $f(\omega;\boldsymbol{\gamma})$ or a sample version of it for given time series data. If $\sigma^2(\boldsymbol{\gamma})$ denotes the variance of $\boldsymbol{\gamma}^\top \boldsymbol{Y}_t$, then $\boldsymbol{\gamma}(\omega)$ is chosen to maximize $f(\omega;\boldsymbol{\gamma})/\sigma^2(\boldsymbol{\gamma})$. The corresponding maximal value, so-called $\lambda(\omega)$, is called the spectral envelope of process X_t. Note that $\lambda(\omega)$ expresses the maximal proportion of the variance that can be explained by the frequency ω, and this maximal proportion is reached if the optimal scaling $\boldsymbol{\gamma}(\omega)$ is used. More details on the computation of $\lambda(\omega)$ and on corresponding sample versions $\widehat{\lambda}(\omega)$ can be found in [2,29,30]. If $\widehat{\lambda}(\omega)$ is plotted against ω, a visual frequency analysis of the categorical time series is possible.

Besides the features introduced above, there are several alternative statistics used to describe categorical series as the so-called pattern-based features, whose main idea relies on discovering certain cyclical behaviors in a given time series [31,32]. An interesting example of such kind of features is given by the so-called motifs, which are short sequences repeating themselves regularly within the time series of interest. It is worth highlighting that motifs have been successfully employed in the context of time series data mining [33,34].

3. Main functions in ctsfeatures

This section is devoted to present the main content of package **ctsfeatures**. First, the datasets available in the package are briefly described, and then the main functions of the package are introduced, including both graphical and analytical tools. It is worth recalling that the analytical functions in **ctsfeatures** require the CTS under consideration to have been generated from stationary processes (see Section 2). Otherwise, misleading conclusions can be reached from the corresponding outputs.

3.1. Available datasets in ctsfeatures

The package **ctsfeatures** includes some CTS datasets that can be used to evaluate different machine learning algorithms or simply for illustrative purposes. Specifically, the available data collections consist of two databases of biological sequences introduced by [15], one database of sequences of sleep stages partially employed by [16], and three synthetic datasets used in [15] for the evaluation of clustering algorithms. All of them are briefly described below.

- **Real datasets**. The first real dataset contains the genome of 32 viruses pertaining to 4 different families. Each genome can be seen as a sequence consisting of the four DNA bases, namely adenine, guanine, thymine and cytosine. Therefore, this database contains 32 CTS with four categories. In addition, we assume the existence of four underlying classes corresponding to the family of each virus. The second database includes 40 protein sequences. As proteins are constituted of 20 different amino acids, each sequence can be originally considered as a CTS with 20 categories. However, we chose to categorize the amino acids into three groups according to their hydrophobicity, which is a common transformation [15,35,36], so that the number of categories is reduced to three. In this case, there exist two different classes in the collection, since each protein is present either in human beings or in variants of COVID-19 virus. The third dataset contains 62 categorical sequences of sleep stages generated from 62 different subjects who took part in a medical study. The participants were monitored during a night and their sleep stages were recorded by experienced technicians every 30 s. The range of each sequence consists of seven categories associated with several sleep stages. One category represents rapid eye movement (REM) sleep, four categories represent different types of nonrapid eye movement (NREM) sleep, and the remaining ones are associated with subjects who were awake or in continuous movement. The subjects under study suffered from nocturnal frontal lobe epilepsy (NFLE) or from REM behavior disorder (RBD), which are two well-known sleep diseases. Thus, these two conditions give rise to the existence of two different classes in the sleep stage database. Data in the first, second and third databases were sourced from the National Center for Biotechnology Information[1] (NCBI), the UniProt Knowledgebase[2] (UniProtKB), and the Research Resource for Complex Physiologic Signals[3] (PhysioNet) websites, respectively.

[1] https://www.ncbi.nlm.nih.gov/genome/browse#!/overview/
[2] https://www.uniprot.org.
[3] https://physionet.org/content/capslpdb/1.0.0/.

Table 2
Summary of the 6 datasets included in **ctsfeatures**.

Dataset	Object	No. series	Series length	No. categories	No. classes
DNA sequences	`GeneticSequences`	32	Variable	4	4
Protein sequences	`ProteinSequences`	40	Variable	3	2
Sleep stages	`SleepStages`	62	Variable	7	2
Synthetic I	`SyntheticData1`	80	600	3	4
Synthetic II	`SyntheticData2`	80	600	3	4
Synthetic III	`SyntheticData3`	80	600	3	4

Table 3
Functions for visualization in **ctsfeatures**.

Graph	Function in ctsfeatures
Time series plot	`plot_cts()`
Rate evolution graph	`plot_reg()`
IFS circle transformation	`plot_ifsct()`
Pattern histogram	`plot_ph()`
Serial dependence plot $(\widehat{v}(l))$	`plot_cramer()`
Serial dependence plot $(\widehat{k}(l))$	`plot_cohen()`
Control chart (cycle length)	`plot_ccc()`
Control chart (marginal distribution)	`plot_mcc()`
Spectral envelope	`plot_se()`

- **Synthetic datasets**. Each one of the synthetic datasets is associated with a particular model of categorical stochastic process, namely Markov Chains (MC), Hidden Markov Models (HMM) and New Discrete ARMA (NDARMA) processes for the first, second and third database, respectively. In all cases, the corresponding database contains 80 series with 3 categories and length $T = 600$, which are split into 4 groups of 20 series each. All series were generated from the same type of process defining the dataset, but the coefficients of the generating model differ between groups. For instance, four distinct transition matrices were used to generate the 80 series forming the first dataset. The specific coefficients were chosen according to Scenarios 1, 2 and 3 in [15] (see Section 3.1 in [15]).

It is worth highlighting that the biological sequences are one of the most common types of CTS arising in practice. In fact, some machine learning algorithms for CTS are specifically designed to deal with this class of series [37,38]. Some papers have also applied different methods for time series data mining to CTS of sleep stages [16,39]. Therefore, it is useful for the user to have collections of these kind of data available through **ctsfeatures**. On the other hand, the distinct classes forming the synthetic databases can be distinguished by means of both the marginal distributions and the serial dependence patterns. This property makes these datasets particularly suitable to evaluate the effectiveness of the features in Table 1 for several machine learning problems. In particular, the usefulness of these features to perform clustering and classification tasks (among others) in these databases is illustrated in Section 4.3. Table 2 provides a summary of the six datasets included in **ctsfeatures**.

3.2. Graphical tools in *ctsfeatures*

The set of visualization tools for CTS available in **ctsfeatures** is presented and properly detailed in this section. Specifically, graphs for exploratory purposes, two types of plots aimed at characterizing serial dependence, and two graphs mainly designed for quality control. Although the corresponding plots are carefully explained throughout this section, an overview of the main functions for visualization is included in Table 3. Notice that, with the aim of providing the user with visually attractive and informative representations, most graphical tools of **ctsfeatures** make use of several functions of the well-known `R` package **ggplot2** [40].

3.2.1. Time series plot

The most common type of graph when dealing with real-valued time series is probably the so-called time series plot, which displays the values of the time series (y-axis) with respect to time (x-axis). A similar representation can be indeed considered for categorical series. As the categories have been coded with integer numbers from 1 to r, the range is $\mathcal{V} = \{1, 2, \ldots, r\}$ and the classical time series plot can be easily depicted for any CTS. However, although this plot can be useful for exploratory purposes, it presents serious problems with nominal time series because a comparison of the different categories in a numerical scale is not possible in this case. Note that this issue no longer exists by analyzing ordinal time series. In fact, as the range of these series exhibits a natural ordering, the corresponding plot is meaningful as long as this ordering is preserved when encoding the categories, i.e., $1 \leq 2 \leq \ldots \leq r$.

The top left panel of Fig. 1 shows the time series plot based on the first 50 observations of the first series in `SyntheticData1`. As this series was generated from a MC, it has a nominal nature, so the corresponding categories were arbitrarily coded by using the integers 1, 2 and 3.

3.2.2. Rate evolution graph

Since the standard time series plot cannot be applied to nominal time series in a completely meaningful way, several alternative tools for a visual analysis of CTS have been introduced in the literature (see [41] for a survey). Although none of these graphs seems to be a perfect substitute for the time series plot in the categorical setting, the so-called rate evolution graph proposed by [42] is one of the most frequently used tools for stationary analysis [7]. Let $\overline{\boldsymbol{Y}}_t = \{\overline{\boldsymbol{Y}}_1, \ldots, \overline{\boldsymbol{Y}}_T\}$ with $\overline{\boldsymbol{Y}}_k = (\overline{Y}_{k,1}, \ldots, \overline{Y}_{k,r})^\top$ such that $\overline{Y}_{k,i} = 1$ if $\overline{X}_k = i$ ($k = 1, \ldots, T$, $i = 1, \ldots, r$) be the binarized time series associated with the T-length categorical series $\overline{X}_t$. Consider the series of cumulated sums given by $\overline{\boldsymbol{C}}_t = \{\overline{\boldsymbol{C}}_1, \ldots, \overline{\boldsymbol{C}}_T\}$ with $\overline{\boldsymbol{C}}_k = \sum_{s=1}^{k} \overline{\boldsymbol{Y}}_s$, $k = 1, \ldots, T$. The rate evolution graph displays a standard time series plot for each one of the components of $\overline{\boldsymbol{C}}_t$ simultaneously in one graph. For each one of the components, the slope of the corresponding curve is an estimate for the associated marginal probability. If the process is stationary, then the curves should be approximately linear with t, while visible violations of linearity indicate nonstationarity. In fact, the rate evolution graph is frequently employed to check the assumption of stationarity before carrying out specific machine learning tasks in CTS databases (see, e.g., Section 6.1 in [15]).

The rate evolution graph corresponding to the first time series in dataset `SyntheticData1` is shown in the bottom left panel of Fig. 1.

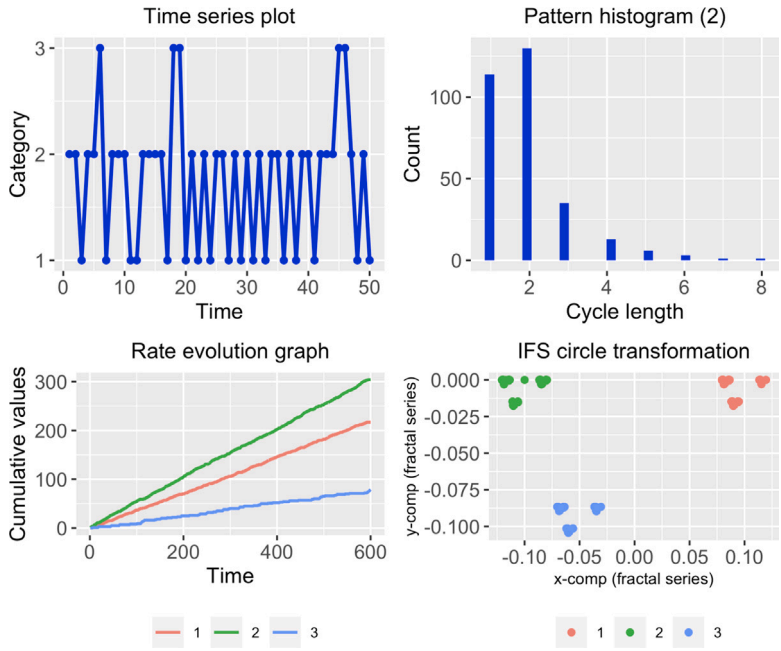

Fig. 1. Some examples of graphical tools available in **ctsfeatures** applied to the first series in dataset `SyntheticData1`: the *time series plot* based on the first 50 records (top left), the *pattern histogram* for the 2nd category (top right), the *rate evolution graph* (bottom left) and the *IFS circle transformation* (bottom right).

3.2.3. Pattern histograms

A categorical series $\overline{X}_t$ can be described by means of the occurrence of certain patterns called cycles. Specifically, a cycle, beginning at time t_1 with $\overline{X}_{t_1} = j$, is closed at time $t_1 + t_2 > t_1$ if and only if the series returns to the value j for the first time after t_1, i.e., if $\overline{X}_{t_1} = j = \overline{X}_{t_1+t_2}$ but $\overline{X}_{t^*} \neq j$ for all $t_1 < t^* < t_1 + t_2$. By definition, a cycle starts at any time, and a previous cycle is closed except the case when the corresponding symbol occurs for the first time. The length of a cycle starting at t_1 and ending at $t_1 + t_2$ is defined as t_2. Note that the cycles can be easily identified by examining the binarized time series. In fact, for a given category $j \in \mathcal{V}$, a cycle of length t_2 starting at time t_1 occurs if and only if $\overline{Y}_{t_1,j} = 1 = \overline{Y}_{t_1+t_2,j}$ but $\overline{Y}_{t^*} = 0$ for all $t_1 < t^* < t_1 + t_2$. The distribution of the length of the cycles existing in a given CTS represents an important statistical feature of the series. Thus, for each category $j \in \mathcal{V}$, [41] suggests to record the corresponding cycles, group them according to their length, and construct an histogram with the corresponding counts. In this way, each series gives rise to r histograms, which are often referred to as pattern histograms. Pattern histograms can be used for several purposes, including model identification, hypothesis testing or anomaly detection, among others.

The top right panel of Fig. 1 contains the pattern histogram concerning the second category of the first series in dataset `SyntheticData1`.

3.2.4. Circle transformation

Other type of exploratory tool for visualizing categorical time series is the so-called IFS circle transformation. This approach was suggested by [43] and is based on iterated function systems (IFS). Although IFS are mainly known for their use to generate fractals, they are frequently applied for the visualization of genetic sequences, a particular type of CTS. The main idea behind the approach proposed by [43] is that an arbitrary categorical series can be appropriately transformed by linear contractions. Afterwards, similar strings are represented by close points in $\mathbb{R}^2$. So pattern analysis can be done visually: the existence and frequency of patterns can be studied by zooming into certain regions of the real plane $\mathbb{R}^2$. The procedure by [43] consists of the following steps:

1. Transform the series range $\mathcal{V}$ by applying the circle transformation, which is a one-to-one mapping $\boldsymbol{\varphi}$ from $\mathcal{V}$ on points of the unit circle in $\mathbb{R}^2$ defined by
$$\boldsymbol{\varphi}(i) = \left(\cos\left(\frac{2\pi(i-1)}{r}\right), \sin\left(\frac{2\pi(i-1)}{r}\right)\right). \tag{10}$$
2. Construct the series $\overline{\boldsymbol{R}}_t = \{\overline{\boldsymbol{R}}_1, \ldots, \overline{\boldsymbol{R}}_T\}$ such that $\overline{\boldsymbol{R}}_k = \boldsymbol{\varphi}(\overline{X}_k)$, $k = 1, \ldots, T$.
3. Generate the fractal series, denoted by $\overline{\boldsymbol{F}}_t = \{\overline{\boldsymbol{F}}_1, \ldots, \overline{\boldsymbol{F}}_T\}$, by means of the following recursion:
$$\overline{\boldsymbol{F}}_k = \alpha \overline{\boldsymbol{F}}_{k-1} + \beta \overline{\boldsymbol{R}}_k, \tag{11}$$
where $\alpha \in (0, 1)$, $\beta > 0$ and $\overline{\boldsymbol{F}}_0$ is an arbitrary initialization from $\mathbb{R}^2$.

A scatterplot of the two-dimensional fractal time series $\overline{\boldsymbol{F}}_t$ (IFS circle transformation) is frequently useful to identify cycles of arbitrary length (see Theorem 4.2 in [41]). In fact, this graph contains frequency information on any subsequence of arbitrary length within it. The corresponding information can be obtained simply by zooming in into interesting regions. Interesting explanations concerning the interpretation of this visualization tool are given in Example 4.3 of [41].

The IFS circle transformation for the first series in dataset `SyntheticData1` is depicted in the bottom right panel of Fig. 1.

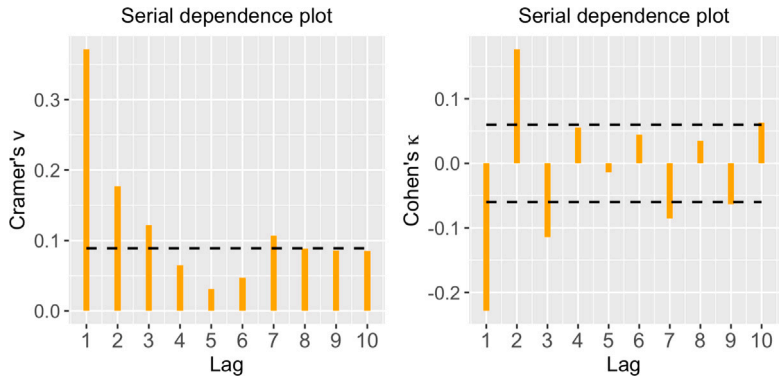

Fig. 2. Serial dependence plots based on $\widehat{v}(l)$ and $\widehat{\kappa}(l)$, with a maximum lag $L = 10$, for the first series in dataset `SyntheticData1`.

3.2.5. Serial dependence plots

When dealing with real-valued time series, a plot of the autocorrelation function (ACF) is usually depicted to identify the lags showing significant correlations and analyze the serial dependence structure. Since this function is not defined in the categorical case (neither nominal nor ordinal), alternative statistical representations are required. For a given CTS, a well-known approach to examine serial dependence at lag $l \in \mathbb{Z}$ consists in analyzing the sample versions of the Cramer's v, $\widehat{v}(l)$, and the Cohen's κ, $\widehat{\kappa}(l)$ (see Table 1). While the former statistic assesses the unsigned dependence, the later measures signed dependence (both dependence notions introduced in Section 2). In both cases, the value 0 is associated with an i.i.d. process. Moreover, the asymptotic distributions of the estimates $\widehat{v}(l)$ and $\widehat{\kappa}(l)$ under the i.i.d. assumption were derived by [28,44], respectively. The corresponding limit distributions are given by

$$T(r-1)\widehat{v}(l)^2 \underset{a}{\sim} \chi^2_{(r-1)^2} \quad \text{and} \quad \sqrt{\frac{T}{\widehat{V}(\boldsymbol{p})}}\left(\widehat{\kappa}(l) + \frac{1}{T}\right) \underset{a}{\sim} N(0,1), \tag{12}$$

where $\widehat{V}(\boldsymbol{p}) = 1 - \dfrac{1 + 2\sum_{i=1}^r \widehat{p}_i^3 - 3\sum_{i=1}^r \widehat{p}_i^2}{(1 - \sum_{i=1}^r \widehat{p}_i^2)^2}$.

The asymptotic results in Eq. (12) are quite useful in practice, since they can be used to test the null hypothesis of serial independence at lag l. In particular, critical values for a given significance level α can be computed, which do not depend on the specific lag. Thus, in both cases, serial dependence plots can be constructed by following a similar approach as the ACF plot in the real-valued case. Specifically, after setting a maximum lag L of interest, the values of $\widehat{v}(l)$ or $\widehat{\kappa}(l)$, for lags l ranging from 1 to L, are simultaneously depicted in one graph. Next, the corresponding critical value is added to the plot by means of a horizontal line. According to Eq. (12), the critical values for an arbitrary significance level α are given by

$$+\sqrt{\frac{\chi^2_{(r-1)^2,1-\alpha}}{T(r-1)}} \quad \text{and} \quad \pm\frac{z_{1-\alpha/2}}{\sqrt{T/V(\widehat{\boldsymbol{p}})}} - \frac{1}{T}, \tag{13}$$

for the plots based on $\widehat{v}(l)$ and $\widehat{\kappa}(l)$, respectively, where $\chi^2_{g,\tau}$ and z_τ denote the respective τ-quantiles of the χ^2 distribution with g degrees of freedom and the standard normal distribution. The described graphs allow us to easily identify the collection of significant lags for a given categorical series. Likewise the autocorrelation plot in the numerical setting, serial dependence plots for CTS can be used for several purposes, including model selection or identification of regular patterns in the series, among others.

For illustrative purposes, the serial dependence plots based on $\widehat{v}(l)$ and $\widehat{\kappa}(l)$ for the first series in dataset `SyntheticData1` are displayed Fig. 2. A maximum lag $L = 10$ was considered in both cases.

3.2.6. Control charts

Techniques of statistical process control (SPC) aim at monitoring and improving production processes. Most popular among them are *control charts*, which help to identify deviations from the state of control, but are also frequently used for exploratory purposes. In the latter context, the calculation of the so-called *control limits* is based on historical data and a hypothetical process model. If some records violate these limits, they are flagged to be *out of control* and possible causes for those deviations are analyzed. In particular, some of these records can be excluded if they are considered as single outliers. Afterwards, revised control limits are determined. However, if the out-of-control points imply a systematic deviation from the model assumptions, then the original model has to be rejected.

As it is stated in [41], it is characteristic for categorical processes that the control statistics C_t does not only depend on the monitoring time t, but also on the concrete value observed at time t, in the sense that the distribution of C_t is influenced by the concrete outcome. Consider, for instance, the case that C_t expresses the length of a cycle ending with $X_t = i$. Then, the corresponding cycle length distribution depends on i, leading to individual control limits LCL_t and UCL_t for C_t. If μ_t denotes the corresponding mean of C_t, we consider the standardized statistics $T_t = T_t^{(L)} + T_t^{(U)}$, where

$$T_t^{(L)} = \min\left(0, \frac{C_t - \mu_t}{|LCL_t - \mu_t|}\right) \quad \text{and} \quad T_t^{(U)} = \max\left(0, \frac{C_t - \mu_t}{|UCL_t - \mu_t|}\right). \tag{14}$$

Thus, an out-of-control alarm is signaled if and only if $T_t < -1$ or $T_t > 1$.

An alternative possibility for controlling categorical processes relies on monitoring their marginal distribution. Specifically, if $\boldsymbol{c} \in [0,1]^r$ and $0 < \lambda < 1$, then

$$\widehat{\boldsymbol{\pi}}_t^{(\lambda)} = \lambda\widehat{\boldsymbol{\pi}}_{t-1}^{(\lambda)} + (1-\lambda)\boldsymbol{Y}_t, \text{ for } t \geq 1, \quad \text{and} \quad \widehat{\boldsymbol{\pi}}_0^{(\lambda)} = \boldsymbol{c}, \tag{15}$$

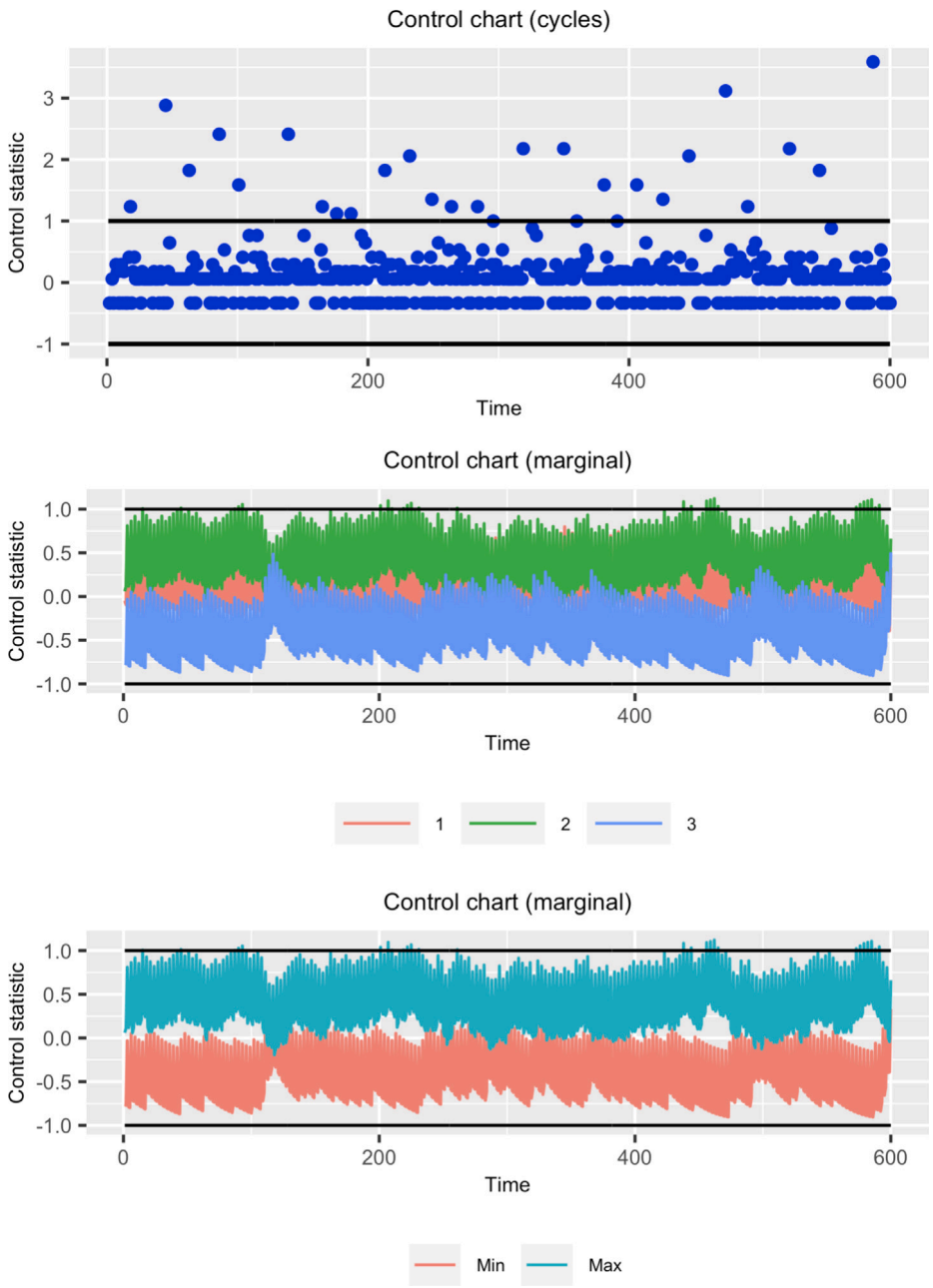

Fig. 3. Control charts associated with Eq. (14) (top panel), Eq. (16) (middle panel) and Eq. (17) (bottom panel) concerning the first time series in dataset `SyntheticData1`.

is called an EWMA estimator of the marginal distribution. If $\boldsymbol{c}$ is set equal to the hypothetical marginal distribution vector, then $\hat{\boldsymbol{\pi}}_t^{(\lambda)}$ is unbiased in the state of control, and its variance $\sigma_{t,i}^2$ is an expression depending on the concrete process model. The exact distribution is difficult to derive, but if λ is chosen large enough, then $k\sigma$-limits will also perform well. So we choose $LCL_i = p_i - k\sigma_{t,i}$ and $UCL_i = p_i + k\sigma_{t,i}$, and statistic in Eq. (14) simplifies to

$$T_{t,i} = \frac{\hat{\pi}_{t,i}^{(\lambda)} - p_i}{k \cdot \sigma_{t,i}}, \tag{16}$$

where $\hat{\pi}_{t,i}^{(\lambda)}$ is the ith component of vector $\hat{\boldsymbol{\pi}}_t^{(\lambda)}$.

Finally, there are some modifications of the EWMA approach for controlling the marginal probabilities. For instance, if the range is very large, then it may be preferable not to plot all statistics $T_{t,i}$ given by the formula Eq. (16) simultaneously, but only the statistics

$$T_t^{\min} = \min_{i \in \mathcal{V}} T_{t,i} \quad \text{and} \quad T_t^{\max} = \max_{i \in \mathcal{V}} T_{t,i}. \tag{17}$$

The panels of Fig. 3 show control charts associated with Eqs. (14) (top), (15) (middle) and (17) (bottom) concerning the first time series in dataset `SyntheticData1`. In all cases, some observations are beyond the upper control limit, thus deserving a careful investigation.

Table 4
Possible values for the argument `type` in function `calculate_features()` along with the corresponding estimates. The complexity associated with the computation of each estimate is indicated in brackets.

Feature	Argument `type`	Feature	Argument `type`
$\widehat{g}$	`'gini_index'` $(O(Tr))$	$\widehat{p}^*(l)$	`'sakoda_measure'` $(O(Tr^2))$
$\widehat{e}$	`'entropy'` $(O(Tr))$	$\widehat{v}(l)$	`'cramers_vi'` $(O(Tr^2))$
$\widehat{c}$	`'chebycheff_dispersion'` $(O(Tr))$	$\widehat{\kappa}(l)$	`'cohens_kappa'` $(O(Tr))$
$\widehat{\tau}(l)$	`'gk_tau'` $(O(Tr^2))$	$\widehat{\Psi}(l)$	`'total_cor'` $(O(Tr^2))$
$\widehat{\lambda}(l)$	`'gk_lambda'` $(O(Tr^2))$	$\widehat{\lambda}(\omega)$	`'spectral_envelope'` $(-)$
$\widehat{u}(l)$	`'uncertainty_coefficient'` $(O(Tr^2))$	$\widehat{\Psi}_1^*(l)$	`'total_mixed_cor'` $(O(Tr))$
$\widehat{X}_n^2(l)$	`'pearson_measure'` $(O(Tr^2))$	$\widehat{\Psi}_2^*(l)$	`'total_mixed_qcor'` $(O(Tr))$
$\widehat{\Phi}^2(l)$	`'phi2_measure'` $(O(Tr^2))$		

As indicated in Table 3, apart from the previously described graphical tools, package **ctsfeatures** also provides the function `plot_se()` to represent the spectral envelope of a given CTS (see Section 2).

3.3. Functions for feature extraction in ctsfeatures

Two main functions are available to compute well-known statistical quantities providing information on marginal and serial properties of CTS. The first function, `calculate_features()`, allows to calculate the estimated features presented in Section 2. The specific feature one wishes to calculate is indicated to the function by means of the argument `type`. The possible values for this argument according to the different estimated quantities are given in Table 4. The second function, `calculate_subfeatures()`, returns the individual components of the corresponding features, and also uses the argument `type`. For instance, `calculate_features(type='cohens_kappa')` computes by default the value of $\widehat{\kappa}(l)$ for a given lag l, which must be properly given as input. However, by running `calculate_subfeatures(type='cohens_kappa')`, a vector formed by the estimates of $\dfrac{p_{jj}(l)-p_j^2}{1-\sum_{i=1}^r p_i^2}$, $j=1,\ldots,r$, is returned. Note that the latter function is usually more useful than the former in machine learning tasks, since it provides a more meaningful description of the dependence structures of the CTS. In Table 4, the computational complexity of the function `calculate_features()` according to the different values for `type` is also provided in brackets. The only exception corresponds to the case of `type='spectral_envelope'`, since the corresponding computation is based on a function of the R package **astsa** [45]. Note that all the serial features introduced in Table 1 for a one-dimensional process have an associated complexity of $O(Tr^2)$ except for the Cohen's κ, which can be computed with a complexity of $O(Tr)$.

Apart from a structural description of the categorical time series, a shape-based characterization is also helpful. With this purpose in mind, **ctsfeatures** also includes the function `calculate_motifs()`, which returns the relative frequencies of words of a certain length. This way, users can identify the existence of relevant shape-based patterns and analyze levels of similarity between several CTS based on their common substructures, which is interesting in different data mining tasks [33,34]. The computational complexity of this function is $O(Tr^W)$, with W being the length of the corresponding word.

4. Using ctsfeatures: Generalities and illustrative examples

This section is devoted to illustrate the use of package **ctsfeatures**. First, we give some general considerations about the package and, next, we provide some examples concerning the use of several functions for visualization and feature extraction.

4.1. Some generalities about ctsfeatures

Most functions in **ctsfeatures** take as input a T-length CTS, represented through an object of class `tsibble` (see package **tsibble** [46]). `tsibble` objects are data structures specifically designed to store time series. Specifically, they must contain an `index` column giving information about the times the observations were recorded. The `tsibble` objects given as input to **ctsfeatures** functions must include the corresponding categorical sequences in a column named `Value`, of class `factor`, the standard R class for categorical objects. Factors have associated a vector of character strings, called `levels`, where the categories forming the range of the CTS are stored and arranged in a given order, usually specified by the user.

The output of the graphical functions listed in Table 3 is simply the requested plot, which can be customized to some extent. Functions `calculate_features()` and `calculate_subfeatures()` return by default the corresponding estimate or its individual components, respectively. It is worth remarking that most functions in **ctsfeatures** require a proper specification of the underlying categories of the series, which is done by means of the column `Value` in the corresponding `tsibble` object. This can be very useful in several situations, including the case in which a particular realization lacks some of the values in the original range. For instance, if one specific realization lacks the rth category, then the function `calculate_subfeatures(type='entropy')` can return the correct output $(\widehat{p}_1 \ln \widehat{p}_1, \widehat{p}_2 \ln \widehat{p}_2, \ldots, \widehat{p}_{r-1} \ln \widehat{p}_{r-1}, 0)$ if the original range is properly provided. Therefore, when analyzing such a series, one could circumvent the lack of the corresponding category by properly using the column `Value`. In addition, this column allows the user to specify the desired order for a particular analysis, which can be helpful in many applications. For instance, when dealing with DNA sequences, the usual convention consists of considering the DNA base adenine as the first one.

As with the individual time series that are given as input to most functions, every database included in **ctsfeatures** is stored as a `tsibble` with the name indicated in the second column of Table 2 and contains four columns. One of them is the column `Value`, previously introduced. The remaining three columns are described below.

- The column called `Series` contains integer values ranging from 1 to the number of series in the dataset, and allows to identify the time series.
- The column called `Time` includes the temporal information associated with each observation.

- The column called `Class` contains integer values indicating the class labels associated with each time series.

Let's visualize the first twenty observations of one series in the dataset `GeneticSequences`.

```
> library(ctsfeatures)
> series <- GeneticSequences[which(GeneticSequences$Series==1),]
> series$Value[1:20]
[1] a t g g c c c a a g c a c a a a t t c t
Levels: a c g t
```

In this series, the DNA bases, adenine, guanine, thymine and cytosine, were coded with the letters 'a', 'g', 't' and 'c', respectively. In addition, the categories were alphabetically ordered. Indeed, a different order could be specified, but one should be always aware of the considered order when analyzing the output of several functions (see Sections 4.2 and 4.3). Concerning the column `Class` in `GeneticSequences`, this column takes four different values (1, 2, 3 or 4) depending on the family of each virus.

```
> unique(GeneticSequences$Class)
[1] 1 2 3 4
```

Note that the integer values in the previous output are not related to the different categories in the range of the series under study, but they just represent the different classes existing in the data collection `GeneticSequences` considering a context of supervised classification.

4.2. Visualizing categorical time series and testing for serial dependence

The functions described in Section 3.2 implementing graphical tools allow the user to construct valuable plots on the performance and dependence structure of a categorical time series. The classical plot showing the profile of the series is obtained with the function `plot_cts()`. For instance, a plot based on the first 50 observations of the first series in dataset `SyntheticData1` is generated as follows.

```
> series <- SyntheticData1[which(SyntheticData1$Series==1)[1 : 50],]
> plot_cts(series)
```

Note that the time series in dataset `SyntheticData1` are nominal, taking values on three attributes coded with integers from 1 to 3. The desired order for the categorical range is indicated through the column `Value`. The corresponding output is depicted in the top left panel of Fig. 1 and valuable information can be extracted from this graph. For instance, it seems that the third category has a much lower marginal probability than the remaining. In addition, one could try to derive some conclusions about the serial dependence structure of the series. However, as stated in Section 3.2.1, the time series plot can be misleading when visualizing nominal time series. In fact, the consideration of a numerical scale in the y-axis implicitly assumes an underlying ordering, which does not exist in the nominal setting. One of the most well-known proposals for the visual analysis of nominal time series is the rate evolution graph (see Section 3.2.2), which is obtained for the first series in `SyntheticData1` by executing

```
> series <- SyntheticData1[which(SyntheticData1$Series==1),]
> plot_reg(series)
```

The above code produces the plot shown in the bottom left panel of Fig. 1. This representation indicates a stationary behavior of the examined time series (at least concerning the marginal distribution), since the three curves exhibit an approximately linear behavior. The slopes of these curves give estimates of the corresponding marginal probabilities. Note that the slope of the blue line is substantially lower than the slopes of their counterparts, thus indicating that the third category appears with much lower frequency than the remaining ones, which is consistent with the time series plot in the top left panel of Fig. 1.

While the rate evolution graph gives information about the marginal distribution, some alternative graphical representations are useful to analyze the existence of certain serial patterns, such as cycles. One of them is the pattern histogram (see Section 3.2.3), which represents counts of the cycles associated with a given category $i \in \mathcal{V}$ after grouping them in accordance with their length. Pattern histograms can be directly generated by using the function `plot_ph()`, whose main arguments are the corresponding CTS and the category for which we wish to analyze the cycles. In the following code, we indicate `category=2`.

```
> plot_ph(series, category = 2)
```

The generated pattern histogram is shown in the top right panel of Fig. 1. It is observed that the second category frequently repeats itself either immediately or after one step, as the cycles of lengths 1 and 2 are the most common. Note that large counts for these cycles are totally related to the fact that the second category is the one displaying the maximum marginal probability as seen in the rate evolution graph. Similar analyses could be carried out by considering the pattern histograms of the remaining two categories.

Another interesting graph to examine the cyclical behavior of a categorical series is the circle transformation (see Section 3.2.4). Frequency information concerning any subsequence in the original CTS can be extracted from this tool. Let's construct the circle transformation of the series we are dealing with throughout this section. To this aim, we use the function `plot_ifsct()`. The parameters α and β involved in the construction of the fractal time series in Eq. (11) must be given as input to this function.

```
> ct <- plot_ifsct(series, alpha = 0.17, beta = 0.10)
> ct
```

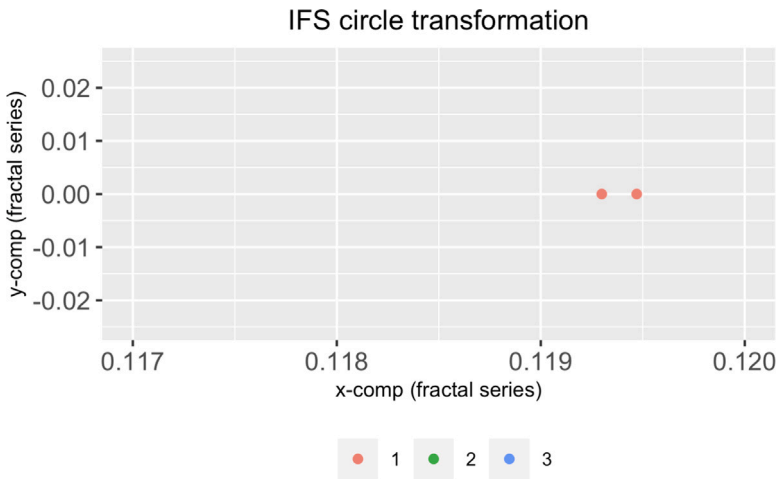

Fig. 4. Region of interest (for subsequences of the form '111') concerning the circle transformation in the bottom right panel of Fig. 1.

For illustrative purposes, we set the values $\alpha = 0.17$ and $\beta = 0.10$. The corresponding plot is provided in the bottom right panel of Fig. 1. This graph is of limited use to perform a static analysis, but it is very helpful as an interactive exploratory tool, since the circles of any level are always organized in the same way. For instance, a subsequence starting with '1' is always at the top of its parent circle. In fact, by navigating appropriately within the graph, one could determine the frequency of any arbitrary string in the original series. Thus, to obtain the number of subsequences of the form '111', we should first zoom into the upper right part within the circle of red points. This results in another circle whose upper right part (which can be accessed again via zooming) contains as many points as subsequences of the desired type. The command `coord_cartesian()` of the R package **ggplot2** [40] provides an easy way to navigate within a given graph.

```
> ct + coord_cartesian(x = c(0.117, 0.120), y = c(-0.025, 0.025))
```

Fig. 4 displays the resulting graph, which contains only two points. Therefore, we can conclude that only two subsequences of the form '111' exist in the first series in dataset `SyntheticData1`. The frequency of any subsequence of arbitrary length could be obtained analogously. An interesting analysis regarding the interpretation of the graph resulting from the circle transformation can be seen in [41] (Example 4.3).

The previous graphs are frequently used for standard exploratory data analysis. However, **ctsfeatures** also contains some tools to carry out more specific tasks. One of such tools are the serial dependence plots described in Section 3.2.5, which can be obtained by means of the functions `plot_cramer()` and `plot_cohen()`. For instance, the plots shown in the left and right panels of Fig. 2 correspond to the values $\hat{\upsilon}(l)$ and $\hat{\kappa}(l)$, respectively, for the first series in `SyntheticData1`, and were obtained by running the code lines below.

```
> plot_1 <- plot_cramer(series)
> plot_2 <- plot_cohen(series)
```

By default, both functions consider lags from 1 to 10 (argument `max_lag`) and a significance level $\alpha = 0.05$ for the tests in Eqs (12) and (13) (argument `alpha`). As the standard autocorrelation plots, the corresponding estimates are displayed in a sequential order, with dashed lines indicating the critical values for the associated tests. In this case, the serial dependence plots clearly exhibit significant dependence at lags 1, 2 and 3. Moreover, dependence at lag 7 could also be considered significant, but this may be due to chance since multiple tests are simultaneously carried out. Indeed, the dependence structure suggested by both graphs is expected, since the considered series was generated from a MC. Certainly, a realization of a MC is expected to show a decreasing trend with respect to the degree of serial dependence. Specifically, the level of dependence is maximum at lag 1, then progressively decreases for further lags until it is no longer significant. Besides the dependence plots, functions `plot_cramer()` and `plot_cohen()` also produce numerical outputs. For instance, the corresponding p-values can be obtained by using the argument `plot = FALSE`.

```
> plot_1 <- plot_cramer(series, plot = FALSE)
> plot_2 <- plot_cohen(series, plot = FALSE)
> round(plot_1$p_values, 2)
 [1] 0.00 0.00 0.00 0.28 0.88 0.61 0.01 0.05 0.07 0.07
> round(plot_2$p_values, 2)
 [1] 0.00 0.00 0.00 0.07 0.62 0.15 0.01 0.24 0.04 0.04
```

The p-values in these outcomes corroborate that both the Cramer's υ and the Cohen's κ are significantly nonnull at the first three lags, thus confirming the existence of significant serial dependence at these lags. Note that the p-value associated with lag 7 is rather low in both cases (0.01) and $\kappa(9)$ and $\kappa(10)$ also differ from zero in a significant way for $\alpha = 0.05$. However, each set of p-values should be properly adjusted to handle random rejections of the null hypothesis that can arise in a multiple testing context. For instance, the well-known Holm's method, which controls the family-wise error rate at a pre-specified α-level, could be applied to the p-values associated with the $\hat{\kappa}(l)$-based test by executing

```
> p.adjust(round(plot_2$p_values, 2), method = 'holm')
[1] 0.00 0.00 0.00 0.28 0.62 0.45 0.07 0.48 0.24 0.24
```

According to the corrected p-values, the values of $\hat{\kappa}(l)$ for $l = 1, 2$ and 3 are still significant, but the null hypothesis of serial independence at lags 7, 9 and 10 cannot be now rejected.

4.3. Performing data mining tasks with ctsfeatures

Jointly used with external functions, **ctsfeatures** becomes a versatile and helpful tool to carry out different machine learning tasks involving categorical series. In this section, for illustrative purposes, we focus our attention on three well-known problems, namely classification, clustering and outlier detection. However, it is worth emphasizing that package **ctsfeatures** is not limited to the mentioned tasks and can be used for multiple purposes beyond them, e.g., model selection and inference or regression, among others.

4.3.1. Performing CTS classification

Firstly, we show how the output of the functions in Table 4 can be used to perform feature-based classification. We illustrate this approach by considering the data collection `GeneticSequences`, which contains the genome of 32 viruses pertaining to 4 different families (class labels). Using the function `calculate_features()`, we create the matrix `feature_dataset` formed by the values $\widehat{g}$ and $\widehat{v}(1)$ extracted from each one of the 32 series as follows.

```
> l_1 <- list()
> l_2 <- list()
> classes <- numeric()

> for (i in 1 :32) {

+ ind <- which(GeneticSequences$Series==i)
+ l_1[[i]] <- calculate_features(GeneticSequences[ind,],type='gini_index')
+ l_2[[i]] <- calculate_features(GeneticSequences[ind,],type='cramers_vi')
+ classes[i] <- unique(GeneticSequences[ind,]$Class)

+ }

> feature_dataset <- cbind(unlist(l_1), unlist(l_2))
> head(feature_dataset, 3)
      [,1]    [,2]
[1,] 0.9975936 0.11578926
[2,] 0.9971419 0.11874799
[3,] 0.9963633 0.11308114
```

The ith row of `feature_dataset` contains estimated values characterizing the marginal and serial behavior of the ith CTS in the dataset. Therefore, several traditional classification algorithms can be applied to the matrix `feature_dataset` by means of the `R` package **caret** [47]. Package **caret** requires the dataset of features to be an object of class `data.frame` whose last column must provide the class labels of the elements and be named `'Class'`. Thus, as a preliminary step, we create `df_feature_dataset`, a version of `feature_dataset` properly arranged to be used as input to **caret** functions, by means of the following chunk of code.

```
> df_feature_dataset <- data.frame(cbind(feature_dataset, classes))
> colnames(df_feature_dataset)[3] <- 'Class'
> df_feature_dataset[,3] <- factor(df_feature_dataset[,3])
```

The function `train()` allows to fit several classifiers to the corresponding dataset, while the selected algorithm can be evaluated, for instance, by Leave-One-Out Cross-Validation (LOOCV). A grid search in the hyperparameter space of the corresponding classifier is performed by default. First, we consider a standard classifier based on k Nearest Neighbors (kNN) by using `method = 'knn'` as input parameter. By means of the command `trControl()`, we define LOOCV as evaluation protocol.

```
> library(caret)
> train_control <- trainControl(method = 'LOOCV')
> set.seed(123)
> model_kNN <- train(Class~., data = df_feature_dataset,
+           trControl = train_control, method = 'knn')
```

The object `model_kNN` contains the fitted model and the evaluation results, among others. The reached accuracy can be accessed as follows.

```
> max(model_kNN$results$Accuracy)
[1] 0.78125
```

The kNN classifier achieves an accuracy of 0.78 in the dataset of genetic sequences. Next, we study the performance of the random forest. To this aim, we need to set `method = 'rf'`.

```
> set.seed(123)
> model_RF <- train(Class~., data = df_feature_dataset,
+          trControl = train_control, method = 'rf')
> max(model_RF$results$Accuracy)
[1] 0.90625
```

The random forest obtains a higher accuracy, around 0.91. Any other classifier can be evaluated in a similar way. Note that, in the previous analyses, the serial dependence structure of the series was summarized by means of a single estimate, $\widehat{v}(1)$. However, a more meaningful description of the CTS could be provided by employing the individual features involved in the definition of $\widehat{v}(l)$, that is, the quantities $\frac{\left(\widehat{p}_{ij}(l)-\widehat{p}_i\widehat{p}_j\right)^2}{\widehat{p}_i\widehat{p}_j}$, $i,j=1,\ldots,r$. This way, the degree of deviation from serial independence is considered separately for each pair (i,j). These features can be directly obtained by using the function `calculate_subfeatures()`. To perform the classification by considering these new features, we should proceed as follows.

```
> for (i in 1 :32) {

+ ind <- which(GeneticSequences$Series==i)
+ l_2[[i]]<-calculate_subfeatures(GeneticSequences[ind,],
+ lag = 1, type='cramers_vi')

+ }

> new_feature_dataset <- cbind(unlist(l_1), matrix(unlist(l_2),
+                ncol = 16, byrow = T))
> dim(new_feature_dataset)
[1] 32 17
> df_new_feature_dataset <- data.frame(cbind(new_feature_dataset,
+ classes))
> colnames(df_new_feature_dataset)[18] <- 'Class'
> df_new_feature_dataset[,18] <- factor(df_new_feature_dataset[,18])
> set.seed(123)
> model_kNN <- train(Class~., data = df_new_feature_dataset,
+           trControl = train_control, method = 'knn')
> max(model_kNN$results$Accuracy)
[1] 0.84375
> set.seed(123)
> model_RF <- train(Class~., data = df_new_feature_dataset,
+          trControl = train_control, method = 'rf')
> max(model_RF$results$Accuracy)
[1] 0.90625
```

While the kNN classifier improves its accuracy when the new features are considered, the random forest exhibits exactly the same effectiveness for both sets of features.

Classification of CTS is often a challenging problem due to the complex nature of temporal data. If a feature-based approach is considered, the quality of the classification procedure strongly depends on the capability of the selected features to distinguish the underlying classes (as it has just been illustrated). In this context, trial and error (e.g., via LOOCV) is a common way of determining a suitable set of features. To illustrate the previous remarks, we decided to analyze the quality of the features based on pattern histograms (see Sections 3.2.3 and 4.2). Specifically, for each time series, we extract the counts of the cycle lengths for the category 'a' (adenine). In particular, only the counts associated with cycles of lengths less than or equal to ten are considered. The following chunk of code creates the desired dataset.

```
> l_counts <- list()

> for (i in 1 :32) {

+ ind <- which(GeneticSequences$Series==i)
+ l_counts[[i]] <- plot_ph(GeneticSequences[ind,],
+ category = 'a', plot= F)[,2][1:10]

+ }

> feature_counts <- matrix(unlist(l_counts), nrow = 32, byrow = T)
> head(feature_counts, 3)
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  111   81   68   43   30   38   26   13   10    8
[2,]   99   77   65   48   31   31   25   13   17   11
[3,]  110   79   76   38   35   27   23   12   11    7
```

Note that the argument `plot = FALSE` in the function `plot_ph()` allows to obtain the counts. Let's analyze the accuracy of the previously considered classifiers (kNN and random forest) with these new descriptive features.

```
> df_feature_counts <- data.frame(cbind(feature_counts, classes))
> colnames(df_feature_counts)[11] <- 'Class'
> df_feature_counts[,11] <- factor(df_feature_counts[,11])
> set.seed(123)
```

```
> model_kNN <- train(Class~., data = df_feature_counts,
+           trControl = train_control, method = 'knn')
> max(model_kNN$results$Accuracy)
[1] 0.78125
> set.seed(123)
> model_RF <- train(Class~., data = df_feature_counts,
+          trControl = train_control, method = 'rf')
> max(model_RF$results$Accuracy)
[1] 0.8125
```

Although both methods show a worse performance than in previous analyses, they exhibit a rather high accuracy, thus indicating a great ability of the considered features (frequencies of cycle lengths for the category 'a') to identify the underlying families of viruses. Indeed, we could examine the classification effectiveness of alternative sets of features (e.g., those involved in the definition of the estimates in Table 1) by proceeding in exactly the same way.

4.3.2. Performing CTS clustering

The package **ctsfeatures** also provides an excellent framework to carry out clustering of categorical sequences. Let's consider the dataset `SyntheticData1` and assume that the clustering structure is governed by the similarity between underlying models. In other terms, the ground truth is given by the 4 groups involving the 20 series from the same generating process. We wish to perform clustering and, according to our criterion, the clustering effectiveness of each algorithm must be measured by comparing the experimental solution with the true partition defined by these four groups.

In cluster analysis, distances between data objects play an essential role. In our case, a suitable metric should take low values for pairs of series coming from the same stochastic process, and high values otherwise. A classical exploratory step to shed light on the quality of a particular metric consists of constructing a two-dimensional scaling (2DS) based on the corresponding pairwise distance 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). For instance, we are going to construct the 2DS for dataset `SyntheticData1` by using two specific metrics between CTS proposed by [15] and denoted by d_{CC} and d_B. More specifically, given two CTS, $\overline{X}_t^{(1)}$ and $\overline{X}_t^{(2)}$, the distances are defined as follows.

$$d_{CC}\left(\overline{X}_t^{(1)}, \overline{X}_t^{(2)}\right) = \sum_{l=1}^{L}\sum_{i,j=1}^{r}\left(\frac{\left(\widehat{p}_{ij}^{(1)}(l) - \widehat{p}_i^{(1)}\widehat{p}_j^{(1)}\right)^2}{\widehat{p}_i^{(1)}\widehat{p}_j^{(1)}} - \frac{\left(\widehat{p}_{ij}^{(2)}(l) - \widehat{p}_i^{(2)}\widehat{p}_j^{(2)}\right)^2}{\widehat{p}_i^{(2)}\widehat{p}_j^{(2)}}\right)^2 + \sum_{l=1}^{L}\sum_{i=1}^{r}\left(\frac{\widehat{p}_{ii}^{(1)}(l) - \widehat{p}_i^{(1)}\widehat{p}_i^{(1)}}{1 - \sum_{j=1}^{r}\widehat{p}_j^{(1)}\widehat{p}_j^{(1)}} - \frac{\widehat{p}_{ii}^{(2)}(l) - \widehat{p}_i^{(2)}\widehat{p}_i^{(2)}}{1 - \sum_{j=1}^{r}\widehat{p}_j^{(2)}\widehat{p}_j^{(2)}}\right)^2 + \sum_{i=1}^{r}\left(\widehat{p}_i^{(1)} - \widehat{p}_i^{(2)}\right)^2, \tag{18}$$

$$d_B\left(\overline{X}_t^{(1)}, \overline{X}_t^{(2)}\right) = \sum_{l=1}^{L}\sum_{i,j=1}^{r}\left(\widehat{\psi}_{ij}^{(1)}(l) - \widehat{\psi}_{ij}^{(2)}(l)\right)^2 + \sum_{i=1}^{r}\left(\widehat{p}_i^{(1)} - \widehat{p}_i^{(2)}\right)^2, \tag{19}$$

where L denotes the largest lag and superscripts (1) and (2) indicate that the corresponding estimates are based on the realizations $\overline{X}_t^{(1)}$ and $\overline{X}_t^{(2)}$, respectively. Note that both dissimilarities assess discrepancies between the marginal distributions (last term) and the serial dependence structures (remaining terms) of both series. Therefore, they seem appropriate to group the CTS of a given collection in terms of underlying stochastic processes.

Let's first create the datasets `dataset_dcc` and `dataset_db` with the features required to compute d_{CC} and d_B, respectively. As the series in `SyntheticData1` were generated from MC processes, we consider only one lag to construct the distance, i.e., we set $L = 1$ (default option).

```
> list_1_dcc <- list()
> list_2_dcc <- list()
> list_3_dcc <- list()
> classes <- numeric()

> for (i in 1 : 80) {

+ ind <- which(SyntheticData1$Series==i)
+ list_1_dcc[[i]] <- calculate_subfeatures(SyntheticData1[ind,],
+ type='cramers_vi')
+ list_2_dcc[[i]] <- calculate_subfeatures(SyntheticData1[ind,],
+ type='cohens_kappa')
+ list_3_dcc[[i]] <- marginal_probabilities(SyntheticData1[ind,])
 classes[i]<- unique(SyntheticData1[ind,]$Class)

+ }

> dataset_1_dcc <- matrix(unlist(list_1_dcc), nrow = 80, byrow = T)
> dataset_2_dcc <- matrix(unlist(list_2_dcc), nrow = 80, byrow = T)
> dataset_3_dcc <- matrix(unlist(list_3_dcc), nrow = 80, byrow = T)
> dataset_dcc <- cbind(dataset_1_dcc, dataset_2_dcc, dataset_3_dcc)

> list_1_db <- list()
```

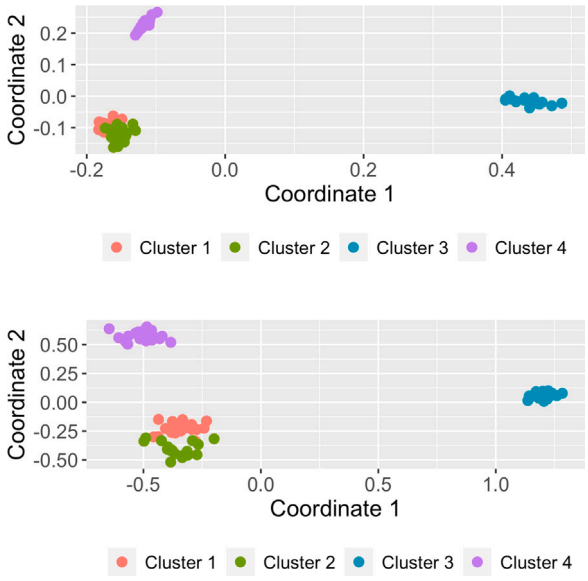

Fig. 5. Two-dimensional scaling planes based on distances d_{CC} (top panel) and d_B (bottom panel) for the 80 series in the dataset `SyntheticData1`.

```
> list_2_db <- list()

> for (i in 1 : 80) {

+ ind <- which(SyntheticData1$Series==i)
+ list_1_db[[i]] <- calculate_subfeatures(SyntheticData1[ind,],
+ type='total_correlation')
+ list_2_db[[i]] <- marginal_probabilities(SyntheticData1[ind,])

+ }

> dataset_1_db <- matrix(unlist(list_1_db), nrow = 80, byrow = T)
> dataset_2_db <- matrix(unlist(list_2_db), nrow = 80, byrow = T)
> dataset_db <- cbind(dataset_1_db, dataset_2_db)
```

The 2DS planes can be built using the function `plot_2d_scaling()` of the **R** package **mlmts** [48], which takes as input a pairwise dissimilarity matrix.

```
> library(mlmts)
>
> distance_matrix_dcc <- dist(dataset_dcc)
> plot_dcc <- plot_2d_scaling(distance_matrix_dcc,
+    cluster_labels = classes)$plot
> distance_matrix_db <- dist(dataset_db)
> plot_db <- plot_2d_scaling(distance_matrix_db,
+    cluster_labels = classes)$plot
```

The resulting plots are shown in Fig. 5. In both cases, the points were colored according to the true partition defined by the generating models. For it, we had to include the argument `cluster_labels` in the function `plot_2d_scaling()`. This option is indeed useful to examine whether a specific metric is appropriate when the true class labels are known. The 2DS planes reveal that both metrics are able to identify the underlying structure rather accurately. However, there are two specific groups of CTS (the ones represented by red and green points) exhibiting a certain degree of overlap in both plots, which suggests a high level of similarity between the corresponding generating processes. This can be corroborated by observing that the coefficients in the transition matrices of these processes are certainly very similar (see Scenario 1 in Section 3.1 of [15]).

To evaluate the clustering accuracy of both metrics, we consider the popular Partitioning Around Medoids (PAM) algorithm, which is implemented in R through the function `pam()` of package **cluster** [49]. This function needs the pairwise distance matrix and the number of clusters. The latter argument is set to 4, since the series in dataset `SyntheticData1` were generated from 4 different MC.

```
> library(cluster)
> set.seed(123)
> clustering_dcc_pam <- pam(distance_matrix_dcc, k = 4)$clustering
> clustering_db_pam <- pam(distance_matrix_db, k = 4)$clustering
```

The vectors `clustering_dcc_pam` and `clustering_db_pam` provide the respective clustering solutions based on both metrics. The quality of both partitions requires to measure their degree of agreement with the ground truth, which can be done by using the Adjusted Rand Index (ARI) [50]. This index can be easily computed by means of the function `external_validation()` of package **ClusterR** [51].

```
> library(ClusterR)
> external_validation(clustering_dcc_pam, classes)
[1] 0.9662323
> external_validation(clustering_db_pam, classes)
[1] 0.9038919
```

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 both metrics d_{CC} and d_B attain great results in this dataset when used with the PAM algorithm. In particular, the partition produced by the former is slightly more similar to the ground truth. Note that the high value of ARI index was already expected from the 2DS plots in Fig. 5.

The popular K-means clustering algorithm can also be executed by using **ctsfeatures** utilities. In this case, we need to employ a dataset of features along with the `kmeans()` function of package **stats** [18].

```
> set.seed(123)
> clustering_dcc_kmeans <- kmeans(dataset_dcc, c = 4)$cluster
> external_validation(clustering_dcc_kmeans, classes)
[1] 1
> set.seed(123)
> clustering_db_kmeans <- kmeans(dataset_db, c = 4)$cluster
> external_validation(clustering_db_kmeans, classes)
[1] 0.9341667
```

In this example, slightly better results are obtained when the K-means algorithm is employed, although the differences with respect to the ARI values produced by the PAM algorithm do not seem statistically significant. The performance of alternative dissimilarities or collections of features concerning a proper identification of the underlying clustering structure could be determined by following the same steps as in the previous experiments.

4.3.3. Performing outlier detection in CTS datasets

Other challenging issue when analyzing a collection of CTS is to detect outlier elements. First, it is worth noting that different notions of outliers are considered in the context of temporal data (additive outliers, innovative outliers, and others). Here, we consider the outlying elements to be whole CTS objects. More specifically, an anomalous CTS is assumed to be a series generated from a stochastic process different from those generating the majority of the series in the database.

To illustrate how **ctsfeatures** can be useful to carry out outlier identification, we create a dataset including two atypical elements. To this aim, we consider all the series in `SyntheticData1` along with the first two series in dataset `SyntheticData3`.

```
> tsibble_outliers <- SyntheticData3[1 : 1200,]
> tsibble_outliers$Series <- tsibble_outliers$Series + 80
> library(dplyr)
> data_outliers <- bind_rows(SyntheticData1, tsibble_outliers)
```

The resulting data collection, `data_outliers`, contains 82 CTS. The first 80 CTS can be split into four homogeneous groups of 20 series, but those located into positions 81 and 82 are actually anomalous elements in the collection because they come from a NDARMA model (see Section 3.1).

A distance-based approach to perform anomaly detection consists of obtaining the pairwise distance matrix and proceeding in two steps as follows.

Step 1. For each element, compute the sum of its distances from the remaining objects in the dataset, which is expected to be large for anomalous elements.

Step 2. Sort the quantities computed in Step 1 in decreasing order and reorder the indexes according to this order. The first indexes in this new vector correspond to the most outlying elements, while the last ones to the least outlying elements.

We follow this approach to examine whether the outlying CTS in `data_outliers` can be identified by using the distance d_B given in Eq. (19). First, we construct the pairwise dissimilarity matrix based on this metric for the new dataset.

```
> list_1_db_outl <- list()
> list_2_db_outl <- list()

> for (i in 1 : 82) {

+ ind <- which(data_outliers$Series==i)
+ list_1_db_outl[[i]]<- calculate_subfeatures(data_outliers[ind,],
```

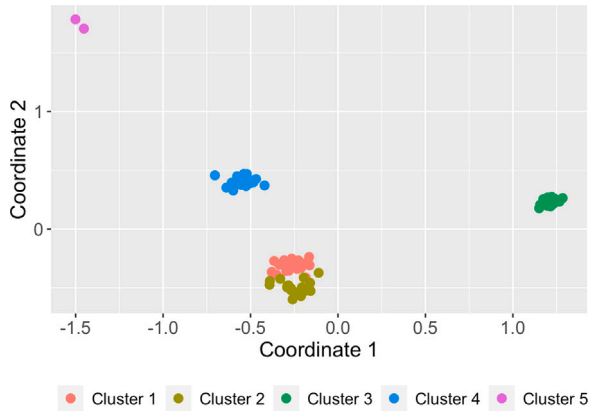

Fig. 6. Two-dimensional scaling plane based on distance d_B for the dataset containing 2 anomalous series.

```
+ type='total_correlation')
+ list_2_db_outl[[i]]<- marginal_probabilities(data_outliers[ind,])

+ }

> data_1_db_outl <- matrix(unlist(list_1_db_outl), nrow = 82, byrow = T)
> data_2_db_outl <- matrix(unlist(list_2_db_outl), nrow = 82, byrow = T)
> data_db_outl <- cbind(data_1_db_outl, data_2_db_outl)
> dist_db_outl <- dist(data_db_outl)
```

Then, we apply the mentioned two-step procedure to the matrix `dist_db_outl` by running

```
> order(colSums(as.matrix(dist_db_outl)), decreasing = T)[1:2]
[1] 81 82
```

The previous output indicates that d_B is able to properly identify the two series generated from an anomalous stochastic process. As an illustrative exercise, let's represent the corresponding 2DS plot for the dataset containing the two outlying CTS by using a different color for these elements.

```
> library(mlmts)
> labels <- c(classes, 5, 5)
> plot_2d_scaling(dist_db_outl, cluster_labels = labels)$plot
```

The resulting graph is shown in Fig. 6. The 2DS configuration is very similar to the one in the bottom panel of Fig. 5 but, this time, two isolated points representing the anomalous series appear on the top left part of the plot. Clearly, 2DS plots can be very useful for outlier identification purposes, since they provide a great deal of information on both the number of potential outliers and their location with respect to the remaining elements in the dataset.

In the previous example, the number of outliers was assumed to be known, which is not realistic in practice. In fact, when dealing with real CTS datasets, one usually needs to determine whether the dataset at hand contains outliers. To that aim, it is often useful to define a measure indicating the outlying nature of each object (see, e.g., [52,53]). In this way, those elements with an extremely large scoring could be identified as outliers. In order to illustrate this approach, we consider the dataset `GeneticSequences` and compute the pairwise distance matrix according to distance d_B.

```
> list_3_db <- list()
> list_4_db <- list()

> for (i in 1 : 32) {

+ ind <- which(GeneticSequences$Series==i)
+ list_3_db[[i]]<- calculate_subfeatures(GeneticSequences[ind,],
+ type='total_correlation')
+ list_4_db[[i]]<- marginal_probabilities(GeneticSequences[ind,])

+ }

> data_3_db <- matrix(unlist(list_3_db), nrow = 32, byrow = T)
> data_4_db <- matrix(unlist(list_4_db), nrow = 32, byrow = T)
> data_db <- cbind(data_3_db, data_4_db)
> dist_db <- dist(data_db)
```

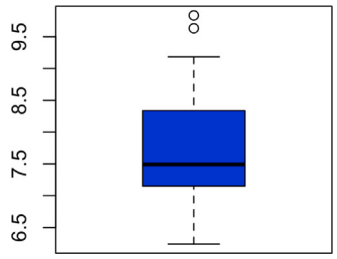

Fig. 7. Boxplot of the outlying scores in dataset `GeneticSequences` based on distance d_B.

As before, the sum of the distances between each series and the remaining ones is computed.

```
> outlier_score <- colSums(as.matrix(dist_db))
```

The vector `outlier_score` contains the sum of the distances for each one of the 32 genetic sequences. Since the ith element of this vector can be seen as a measure of the outlying character of the ith sequence, those genetic sequences associated with extremely large values in this vector are potential outliers. A simple way to detect these series consists of visualizing a boxplot based on the elements of `outlier_score` and checking whether there are points located into the upper part of the graph.

```
> boxplot(outlier_score, range = 1, col = 'blue')
```

The resulting boxplot is shown in Fig. 7 and suggests the existence of two series with abnormally high outlying scores. Hence, these genetic sequences could be considered to be anomalous and their individual properties could be carefully investigated. Note that the prior empirical approach provides an automatic method to determine the number of outliers. Similar analyses could be carried out by considering alternative dissimilarity measures.

5. Concluding remarks

Data mining of time series has experienced a significant growth during the 21st century. Although most of the works focus on real-valued time series, categorical time series have received a great deal of attention during the last decade. The R package **ctsfeatures** is an attempt to provide different functions enabling to compute well-known statistical quantities for categorical series. Besides giving a valuable description about the structure of the time series, the corresponding features can be used as input for traditional machine learning procedures, as clustering, classification and outlier detection algorithms. Furthermore, **ctsfeatures** includes visualization tools and serial dependence plots, thus allowing the user to carry out meaningful exploratory analyses. The main motivation behind the package is that, to the best of our knowledge, no previous R packages are available for a general statistical analysis of categorical series. In fact, the few software tools designed to deal with categorical sequences focus on specific tasks (e.g., computation of dissimilarities), application domains (e.g., sequence analysis), or types of categorical models (e.g., Markov Chains). Package **ctsfeatures** also incorporates two databases of biological sequences, one database of sleep stages, and several synthetic datasets, which can be used for illustrative purposes. In sum, **ctsfeatures** provides rather simple tools, but necessary for standard analyses and of great usefulness to perform more complex tasks as modeling, inference, or forecasting.

A description of the functions available in **ctsfeatures** is given in the first part of this work to make clear the details behind the software and its scope. However, the readers particularly interested in specific tools are encouraged to check the corresponding key references, which are also provided in the article. In the second part, the use of the package is illustrated by means of several examples involving synthetic and real data. This can be seen as a simple overview whose goal is to make the use of **ctsfeatures** as easy and understandable as possible for first-time users.

There are two main ways through which this work can be extended. First, as **ctsfeatures** is under continuous development, we expect to perform frequent updates by incorporating functions for the computation of additional statistical features which are introduced in the future. Second, the package focuses on the analysis of categorical series with nominal range. When the range is ordinal, alternative statistical features can be defined by taking into account the underlying ordering (see [8]). Moreover, in the ordinal setting, these features are more meaningful than their nominal counterparts, since the former ignores the fact that a distance between the different categories can be considered. In this regard, the development of a software tool allowing the computation of statistical features for ordinal time series could be carried out in the future.

CRediT authorship contribution statement

Ángel López-Oriona: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization. **José A. Vilar:** Funding acquisition, Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

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

Acknowledgments

The authors are grateful to the anonymous referees for their comments and suggestions. 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 Universidade da Coruña/CISUG.

References

[1] C.H. Weiß, P.K. Pollett, Binomial autoregressive processes with density-dependent thinning, J. Time Series Anal. 35 (2) (2014) 115–132.
[2] D.S. Stoffer, D.E. Tyler, D.A. Wendt, The spectral envelope and its applications, Stat. Sci. (2000) 224–253.
[3] K. Fokianos, B. Kedem, Regression theory for categorical time series, Stat. Sci. 18 (3) (2003) 357–376.
[4] C.H. Weiß, R. Göb, Measuring serial dependence in categorical time series, AStA-Adv. Stat. Anal. 92 (2008) 71–89.
[5] T. Moysiadis, K. Fokianos, On binary and categorical time series models with feedback, J. Multivariate Anal. 131 (2014) 209–228.
[6] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, D. Haussler, Hidden Markov models in computational biology: Applications to protein modeling, J. Mol. Biol. 235 (5) (1994) 1501–1531.
[7] C.H. Weiß, An Introduction to Discrete-Valued Time Series, John Wiley & Sons, 2018, http://dx.doi.org/10.1002/9781119097013.
[8] C.H. Weiß, Distance-based analysis of ordinal data and ordinal time series, J. Amer. Statist. Assoc. 115 (531) (2020) 1189–1200.
[9] Á. López-Oriona, C.H. Weiß, J.A. Vilar, Two novel distances for ordinal time series and their application to fuzzy clustering, Fuzzy Sets and Systems 468 (2023) 108590.
[10] I. Cadez, D. Heckerman, C. Meek, P. Smyth, S. White, Model-based clustering and visualization of navigation patterns on a web site, Data Min. Knowl. Discov. 7 (2003) 399–424.
[11] M. García-Magariños, J.A. Vilar, A framework for dissimilarity-based partitioning clustering of categorical time series, Data Min. Knowl. Discov. 29 (2) (2015) 466–502.
[12] B.D. Fulcher, Feature-based time-series analysis, in: G. Dong, H. Liu (Eds.), Feature Engineering for Machine Learning and Data Analytics, CRC Press, Boca Raton, 2018.
[13] R. Hyndman, Y. Kang, P. Montero-Manso, M. O'Hara-Wild, T. Talagala, E. Wang, Y. Yang, **tsfeatures**: Time series feature extraction, 2023, URL https://CRAN.R-project.org/package=tsfeatures, R package version 1.1.1.
[14] M. O'Hara-Wild, R. Hyndman, E. Wang, **feasts**: Feature extraction and statistics for time series, 2023, URL https://CRAN.R-project.org/package=feasts, R package version 0.3.1.
[15] Á. López-Oriona, J.A. Vilar, P. D'Urso, Hard and soft clustering of categorical time series based on two novel distances with an application to biological sequences, Inform. Sci. 624 (2023) 467–492.
[16] Z. Li, S.A. Bruce, T. Cai, Interpretable classification of categorical time series using the spectral envelope and optimal scalings, J. Mach. Learn. Res. 23 (299) (2022) 1–31.
[17] M. Horak, S. Chandrasekaran, G. Tobar, NLP based anomaly detection for categorical time series, in: 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science, IRI, IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 27–34.
[18] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2021, URL https://www.R-project.org/.
[19] J.-H. Heine, R.W. Alexandrowicz, M. Stemmler, **confreq**: Configural frequencies analysis using log-linear modeling, 2022, URL https://CRAN.R-project.org/package=confreq, R package version 1.6.1-1.
[20] J.-H. Heine, M. Stemmler, Analysis of categorical data with the R package **confreq**, Psych 3 (3) (2021) 522–541.
[21] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.
[22] A. Gabadinho, G. Ritschard, N.S. Mueller, M. Studer, Analyzing and visualizing state sequences in R with **TraMineR**, J. Stat. Softw. 40 (4) (2011) 1–37.
[23] M. Studer, G. Ritschard, What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures, J. R. Stat. Soc. Ser. A-Stat. Soc. 179 (2) (2016) 481–511.
[24] T.F. Liao, D. Bolano, C. Brzinsky-Fay, B. Cornwell, A.E. Fasang, S. Helske, R. Piccarreta, M. Raab, G. Ritschard, E. Struffolino, et al., Sequence analysis: Its past, present, and future, Soc. Sci. Res. 107 (2022) 102772.
[25] L. StataCorp, Stata Statistical Software: Release 17. 2021, College Station, Texas, United States of America.
[26] B. Halpin, **SADI**: Sequence analysis tools for `Stata`, Stata J. 17 (3) (2017) 546–572.
[27] S.S.D.L. Himmelmann, R package version 1.0.1. https://CRAN.R-project.org/package=HMM,
[28] C.H. Weiß, Serial dependence of NDARMA processes, Comput. Statist. Data Anal. 68 (2013) 213–238.
[29] D.S. Stoffer, D.E. Tyler, A.J. McDougall, Spectral analysis for categorical time series: Scaling and the spectral envelope, Biometrika 80 (3) (1993) 611–622.
[30] R.H. Shumway, D.S. Stoffer, Time Series Analysis and Its Applications, Vol. 3, Springer, 2000.
[31] E. Keogh, S. Lonardi, B.Y.-c. Chiu, Finding surprising patterns in a time series database in linear time and space, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, Association for Computing Machinery, New York, NY, USA, 2002, pp. 550–556.
[32] D. Yankov, E. Keogh, J. Medina, B. Chiu, V. Zordan, Detecting time series motifs under uniform scaling, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 844–853.
[33] A.G. Maletzke, H.D. Lee, G. Enrique, A.P.A. Batista, C.S.R. Coy, J.J. Fagundes, W.F. Chung, Time series classification with motifs and characteristics, in: R. Espin, R.B. Pérez, A. Cobo, J. Marx, Valdés (Eds.), Soft Computing for Business Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 125–138.
[34] E. Ramanujam, S. Padmavathi, Genetic time series motif discovery for time series classification, Int. J. Biomed. Eng. Technol. 31 (1) (2019) 47–63.
[35] I. Dubchak, I. Muchnik, S.R. Holbrook, S.-H. Kim, Prediction of protein folding class using global description of amino acid sequence, Proc. Natl. Acad. Sci. USA 92 (19) (1995) 8700–8704.
[36] I. Dubchak, I. Muchnik, C. Mayor, I. Dralyuk, S.-H. Kim, Recognition of a protein fold in the context of the SCOP classification, Proteins 35 (4) (1999) 401–407.
[37] P.C. FitzGerald, A. Shlyakhtenko, A.A. Mir, C. Vinson, Clustering of DNA sequences in human promoters, Genome Res. 14 (8) (2004) 1562–1574.
[38] N.A. Kassim, A. Abdullah, Classification of DNA sequences using convolutional neural network approach, in: UTM Computing Proceedings Innovations in Computing Technology and Applications, Vol. 2, 2017, pp. 1–6.
[39] M.R. Kirsch, K. Monahan, J. Weng, S. Redline, K.A. Loparo, Entropy-based measures for quantifying sleep-stage transition dynamics: Relationship to sleep fragmentation and daytime sleepiness, IEEE Trans. Biomed. Eng. 59 (3) (2012) 787–796, http://dx.doi.org/10.1109/TBME.2011.2179032.
[40] H. Wickham, **Ggplot2**: Elegant Graphics for Data Analysis, Springer-Verlag New York, 2016, URL https://ggplot2.tidyverse.org.
[41] C.H. Weiß, Visual analysis of categorical time series, Stat. Methodol. 5 (1) (2008) 56–71.
[42] R.L. Ribler, Visualizing Categorical Time Series Data with Applications to Computer and Communications Network Traces (Ph.D. thesis), Virginia Polytechnic Institute and State University, 1997.
[43] C.H. Weiß, R. Göb, Discover patterns in categorical time series using IFS, Comput. Statist. Data Anal. 52 (9) (2008) 4369–4379.
[44] C.H. Weiß, Empirical measures of signed serial dependence in categorical time series, J. Stat. Comput. Simul. 81 (4) (2011) 411–429.
[45] David, N. Poison, **astsa**: Applied statistical time series analysis, 2022, URL https://CRAN.R-project.org/package=astsa, R package version 1.16.
[46] E. Wang, D. Cook, R.J. Hyndman, A new tidy data structure to support exploration and modeling of temporal data, J. Comput. Graph. Stat. 29 (3) (2020) 466–478.
[47] M. Kuhn, Building predictive models in R using the **caret** package, J. Stat. Softw. 28 (5) (2008) 1–26, http://dx.doi.org/10.18637/jss.v028.i05, URL https://www.jstatsoft.org/index.php/jss/article/view/v028i05.
[48] Á. López-Oriona, J.A. Vilar, **mlmts**: Machine learning algorithms for multivariate time series, 2023, URL https://CRAN.R-project.org/package=mlmts, R package version 1.1.1.
[49] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, **cluster**: Cluster analysis basics and extensions, 2021, URL https://CRAN.R-project.org/package=cluster, R package version 2.1.2 — For new features, see the 'Changelog' file (in the package source).
[50] R.J. Campello, A fuzzy extension of the rand index and other related indexes for clustering and classification assessment, Pattern Recognit. Lett. 28 (7) (2007) 833–841.

[51] L. Mouselimis, **ClusterR**: Gaussian mixture models, K-means, mini-batch-kmeans, K-medoids and affinity propagation clustering, 2022, URL https://CRAN.R-project.org/package=ClusterR, R package version 1.2.6.
[52] X. Weng, J. Shen, Detecting outlier samples in multivariate time series dataset, Knowl.-Based Syst. 21 (8) (2008) 807–812.
[53] Á. López-Oriona, J.A. Vilar, Outlier detection for multivariate time series: A functional data approach, Knowl.-Based Syst. 233 (2021) 107527.

Dr. Ángel López-Oriona is a Postdoctoral Fellow at the King Abdullah University of Science and Technology (KAUST), Saudi Arabia. He received his bachelor's degree in Mathematics and his master's degree in Statistics both from the University of Santiago de Compostela, Spain, his master's degree in Big Data Analytics from the European University of Madrid, Spain, and his Ph.D. in Statistics from the University of A Coruña, Spain. During his period as a Ph.D. candidate, he was a visiting researcher in Australia, Germany, Italy, and the United Kingdom. He has authored several papers in JCR journals and participated in many international conferences. His research interests include computational statistics, fuzzy set theory and time series data mining, among others. He loves traveling to exotic places with monkeys, jungles and high-quality beer. Two of Dr. López-Oriona's personal goals are to reach the summit of Mount Kilimanjaro before 2030 and to drive from Madrid to Beijing with his old Volkswagen Passat.

Dr. 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 has 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. Dr. Vilar received the prize to the best Young Spanish Statistician in 1992.