

## Article

# Ordinal Time Series Analysis with the R Package *otsfeatures*

Ángel López-Oriona \*  and José A. Vilar \*

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

\* Correspondence: oriona38@hotmail.com or a.oriona@udc.es (Á.L.-O.); jose.vilarf@udc.es (J.A.V.)

**Abstract:** The 21st century has witnessed a growing interest in the analysis of time series data. While most of the literature on the topic deals with real-valued time series, ordinal time series have typically received much less attention. However, the development of specific analytical tools for the latter objects has substantially increased in recent years. The R package *otsfeatures* attempts to provide a set of simple functions for analyzing ordinal time series. In particular, several commands allowing the extraction of well-known statistical features and the execution of inferential tasks are available for the user. The output of several functions can be employed to perform traditional machine learning tasks including clustering, classification, or outlier detection. *otsfeatures* also incorporates two datasets of financial time series which were used in the literature for clustering purposes, as well as three interesting synthetic databases. The main properties of the package are described and its use is illustrated through several examples. Researchers from a broad variety of disciplines could benefit from the powerful tools provided by *otsfeatures*.

**Keywords:** *otsfeatures*; ordinal time series; feature extraction; cumulative probabilities; R package

**MSC:** 68N01; 62-07



**Citation:** López-Oriona, Á.; Vilar, J.A. Ordinal Time Series Analysis with the R Package *otsfeatures*. *Mathematics* **2023**, *11*, 2565. <https://doi.org/10.3390/math11112565>

Academic Editors: Niansheng Tang and Shen-Ming Lee

Received: 24 April 2023

Revised: 27 May 2023

Accepted: 30 May 2023

Published: 3 June 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Time series data usually arise in a wide variety of disciplines as machine learning, biology, geology, finance and medicine, among many other fields. Typically, most of the works on the analysis of these objects have focused on real-valued time series, while the study of time series with alternative ranges has been given limited attention. However, the latter type of time series naturally appear in several fields when attempting to analyze several phenomena. For instance, weekly counts of new infections with a specific disease in a particular place are often modeled through integer-valued time series [1]. In some contexts, the time series under consideration do not even take numerical values (e.g., temporal records of EEG sleep states for an infant after birth [2]). A comprehensive introduction to the topic of time series with alternative ranges including classical models, recent advances, key references, and specific application areas is provided by [3].

Categorical time series (CTS) are characterized by taking values in a qualitative range consisting of a finite number of categories, which is called ordinal range (if the categories exhibit a natural ordering) or nominal range otherwise. In this paper, the specific case of an ordinal range is considered. Time series fulfilling this condition, frequently referred to as ordinal time series (OTS), pose several challenges to the statistical practitioner. Indeed, dealing with ordered qualitative outcomes implies that some classical analytic tools must be properly adapted. For instance, standard measures of location, dispersion and serial dependence cannot be defined in the same manner as in the real-valued case, but the underlying ordering existing in the series range still allows for a meaningful definition of the corresponding quantities in the ordinal setting. For instance, in [4], a unified approach based on expected distances is proposed to obtain well-interpretable statistical measures

for ordinal series. In addition, sample counterparts of the corresponding measures are introduced and their asymptotic properties are derived.

Ordinal series arise in multiple fields. Some interesting examples include credit ratings of different countries [4] or degree of cloud coverage in different regions [5]. In addition, OTS appear quite naturally in psychology, since temporal measurements in such discipline often originate from ordinal scales, such as Likert questionnaires. For instance, the so-called mood time series of the married couple [6,7] represents the daily mood of a married couple over a period of 144 days, which is recorded on Likert scales. Specifically, the mood measures arise from a questionnaire with 58 items such as “*Right now I feel good*”, being the momentary intensity of emotions rated with answers 1 = *definitely not*, 2 = *not*, 3 = *not really*, 4 = *a little*, 5 = *very much*, and 6 = *extremely*. Such types of time series are naturally considered as ordinal. On the contrary, in many situations, the series under consideration are actually real-valued, but they are treated as ordinal ones because this provides several advantages. For instance, in [8], the gross wage of different individuals is divided into six ordered categories according to the quintiles of the income distribution for each year. As stated by [8], one of the advantages of considering wage categories relies on the fact that no inflation adjustment has to be made. Another illustrative example involves the well-known air quality index (AQI), which presents the status of daily air quality and shows the degree of air pollution in a particular place [9]. The air quality is often classified into six different levels which are determined according to the concentrations of several air pollutants.

While the field of OTS data analysis is still in its early stages, there are already a few interesting works on the topic. In one of the first papers, ref. [10] proposed robust methods of time series analysis which use only comparisons of values and not their actual size. As previously stated, ref. [4] developed an interesting methodology for defining statistical features in the ordinal setting, which is based on expected distances. Later, ref. [11] proposed a family of signed dependence measures for analyzing the behavior of a given OTS. There are also some recent works involving machine learning tasks in the context of ordinal series. For instance, ref. [9] considered different models to forecast the air quality levels in 16 cities of Taiwan. Two novel distances between OTS were proposed in [12] and used to construct effective clustering algorithms. The approaches were applied to datasets of financial time series and interesting conclusions were reached. Previous references highlight the remarkable growth that OTS analysis has recently undergone.

In accordance with previous comments, it is clear that the construction of software tools specifically designed to deal with OTS is crucial. However there exist no software packages in well-known programming languages (e.g., R version 4.1.2 [13], Python version 3.11.1 [14], etc.) aimed at dealing with ordinal series. Moreover, there are only a few libraries focusing on the analysis of ordinal data without a temporal nature, which are mostly written in the R language, but too often restricted to specific statistical procedures. For instance, the package **ordinal** [15] implements cumulative link models for coping with ordinal response variables. Specific functions for generating multivariate ordinal data are provided through package **MultiOrd** [16]. In a purely machine learning context, an innovative computing tool named **ocapis** containing classification algorithms for ordinal data is described in [17]. In addition, the library includes two preprocessing techniques: an instance selector and a feature selector. Note that, although their usefulness is beyond doubt, none of the previously mentioned packages is suitable to execute simple exploratory analyses, a task which should be usually performed before moving on to more sophisticated procedures. In sum, there are currently no software tools allowing to compute classical features for ordinal series.

The goal of this manuscript is to present the R package **otsfeatures**, which includes several functions to compute well-known statistical features for ordinal series. As well as giving valuable information about the behavior of the time series, the corresponding features can be used as input for classical machine learning procedures, as clustering, classification, and outlier detection algorithms. In addition, **otsfeatures** also includes

some commands allowing to perform traditional inferential tasks. The two databases of financial time series described in [12] are also available in the package, along with three synthetic datasets containing OTS which were generated from different underlying stochastic processes. These data collections allow the users to test the commands available in **otsfeatures**. It is worth mentioning that some functions of the package can also be employed to analyze ordinal data without a temporal character.

In summary, the package **otsfeatures** intends to integrate a set of simple but powerful functions for the statistical analysis of OTS into a single framework. The implementation of the package was performed by using the open-source R programming language due to the role of R as the most used programming language for statistical computing. **otsfeatures** is available from the Comprehensive R Archive Network (CRAN) at <https://cran.r-project.org/web/packages/otsfeatures/index.html> (accessed on 15 May 2023).

The rest of the paper is structured as follows. A summary of relevant features to analyze marginal properties and the serial dependence of ordinal series is presented in Section 2. Furthermore, some novel features measuring cross-dependence between ordinal and numerical processes are also introduced. The main functions implemented in **otsfeatures** and the available datasets are presented in Section 3 after providing some background on ordinal series. In Section 4, the functionality of the package is illustrated through several examples considering synthetic data and the financial databases included in **otsfeatures**. In addition, the process of using the output of some functions to carry out traditional data mining tasks is described. Some conclusions are given in Section 5.

## 2. Analyzing Marginal Properties and Serial Dependence of Ordinal Time Series

Let  $\{X_t\}_{t \in \mathbb{Z}}$ ,  $\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$ , be a strictly stationary stochastic process having the ordered categorical range  $\mathcal{S} = \{s_0, \dots, s_n\}$  with  $s_0 < s_1 < \dots < s_n$ . The process  $\{X_t\}_{t \in \mathbb{Z}}$  is often referred to as an ordinal process, while the categories in  $\mathcal{S}$  are frequently called the states. Let  $\{C_t\}_{t \in \mathbb{Z}}$  be the count process with range  $\{0, \dots, n\}$  generating the ordinal process  $\{X_t\}_{t \in \mathbb{Z}}$ , i.e.,  $X_t = s_{C_t}$ . It is well known that the distributional properties of  $\{C_t\}_{t \in \mathbb{Z}}$  (e.g., stationarity) are properly inherited by  $\{X_t\}_{t \in \mathbb{Z}}$  [3]. In particular, the marginal probabilities can be expressed as

$$p_i = P(X_t = s_i) = P(C_t = i), \quad i = 0, \dots, n, \quad (1)$$

while the lagged joint probabilities (for a lag  $l \in \mathbb{Z}$ ) are given by

$$p_{ij}(l) = P(X_t = s_j, X_{t-l} = s_i) = P(C_t = j, C_{t-l} = i), \quad i, j = 0, \dots, n. \quad (2)$$

Note that both the marginal and the joint probabilities are still well defined in the general case of a stationary stochastic process with nominal range, i.e., when no underlying ordering exists in  $\mathcal{S}$ . By contrast, in an ordinal process, one can also consider the corresponding cumulative probabilities, which are defined as

$$\begin{aligned} f_i &= P(X_t \leq s_i) = P(C_t \leq i), \quad i = 0, \dots, n-1, \\ f_{ij}(l) &= P(X_t \leq s_j, X_{t-l} \leq s_i) = P(C_t \leq j, C_{t-l} \leq i), \\ &\quad i, j = 0, \dots, n-1, \quad l \in \mathbb{Z}, \end{aligned} \quad (3)$$

for the marginal and the joint case, respectively.

In practice, the quantities  $p_i$ ,  $p_{ij}(l)$ ,  $f_i$ , and  $f_{ij}(l)$  must be estimated from a  $T$ -length realization of the ordinal process,  $\bar{X}_T = \{\bar{X}_1, \dots, \bar{X}_T\}$ , usually referred to as *ordinal time series* (OTS). Natural estimates of these probabilities are given by

$$\hat{p}_i = \frac{\sum_{k=1}^T I(\bar{X}_k = s_i)}{T}, \quad \hat{p}_{ij}(l) = \frac{\sum_{k=1}^{T-l} I(\bar{X}_k = s_i) I(\bar{X}_{k+l} = s_j)}{T-l}, \quad (4)$$

$$\hat{f}_i = \frac{\sum_{k=1}^T I(\bar{X}_k \leq s_i)}{T}, \quad \hat{f}_{ij}(l) = \frac{\sum_{k=1}^{T-l} I(\bar{X}_k \leq s_i) I(\bar{X}_{k+l} \leq s_j)}{T-l}, \quad (5)$$

where  $I(\cdot)$  denotes the indicator function.

Probabilities  $p_i$ ,  $p_{ij}(l)$ ,  $f_i$ , and  $f_{ij}(l)$  can be used to represent the process  $\{X_t\}_{t \in \mathbb{Z}}$  in terms of marginal and serial dependence patterns. An alternative way of describing a given ordinal process is by means of features measuring classical statistical properties (e.g., centrality, dispersion, etc.) in the ordinal setting. A practical approach to define these quantities consists of considering expected values of some distances between ordinal categories [4]. Specifically, a given distance measure  $d$  defined in  $\mathcal{S} \times \mathcal{S}$  gives rise to specific ordinal features. Three of the most commonly used distances are the so-called Hamming, block, and Euclidean, which are defined as

$$d_H(s_i, s_j) = 1 - \delta_{ij}, \quad d_{o,1}(s_i, s_j) = |i - j| \quad \text{and} \quad d_{o,2}(s_i, s_j) = (i - j)^2, \quad (6)$$

for a pair of states  $s_i$  and  $s_j$ , respectively, where  $\delta_{ij}$  denotes the Kronecker delta. The first six quantities in Table 1 describe the marginal behavior of the process  $X_t$  for a given distance  $d$ . There, the notation DIVC stands for diversity coefficient, which is an approach for measuring dispersion proposed by [18], and  $X_t^1$  and  $X_t^2$  refer to independent copies of  $X_t$ . In addition, the notation  $X_t^r$  was used to define a reflected copy of  $X_t$ , that is, a stochastic process independent of  $X_t$  such that  $P(X_t^r = s_i) = p_{n-i}$ ,  $i = 0, \dots, n$ . Note that the considered location measures pertain to the ordinal set  $\mathcal{S}$ . For the remaining marginal features, some assumptions are needed to obtain the ranges provided in Table 1, where  $d_0^n = d(s_0, s_n)$ . Particularly, for these four measures, we assume that  $d_0^n = \max_{x,y \in \mathcal{S}} d(x, y)$ , a property which is usually referred to as maximization. In addition, for the asymmetry, we require that  $(J - I)D$  is a positive semidefinite matrix, where  $I$  and  $J$  denote the identity and the counteridentity matrices of order  $n + 1$  and  $D = (d_{ij})_{1 \leq i,j \leq n+1}$ , where  $d_{ij} = d(s_{i-1}, s_{j-1})$  is the corresponding pairwise distance matrix. Moreover, for both the asymmetry and the skewness to be reasonable measures, we assume that the distance  $d$  is centrosymmetric, that is,  $d(s_i, s_j) = d(s_{n-i}, s_{n-j})$ ,  $i, j = 0, \dots, n$ . Note that, for a symmetric process (that is, a process  $X_t$  such that  $X_t$  and  $X_t^r$  have the same marginal distribution), then both the asymmetry and the skewness are expected to be zero. It is worth highlighting that, under the required assumptions, the quantities  $\text{disp}_{\text{loc},d}$ ,  $\text{disp}_d$ ,  $\text{asym}_d$  and  $\text{skew}_d$  can be easily normalized to the interval  $[0, 1]$  (or  $[-1, 1]$  in the case of the skewness) by dividing them by  $d_0^n$ .

Estimates of the marginal features in Table 1, denoted by means of the notation  $\widehat{(\cdot)}$ , where  $(\cdot)$  stands for the corresponding measure (e.g.,  $\text{disp}_d$ ), can be obtained by considering estimates of  $E[d(X_t, s_i)]$  ( $i = 0, \dots, n$ ),  $E[d(X_t^1, X_t^2)]$  and  $E[d(X_t, X_t^r)]$  given by

$$\begin{aligned} \widehat{E}[d(X_t, s_i)] &= \frac{1}{T} \sum_{t=1}^T d(\bar{X}_t, s_i), \\ \widehat{E}[d(X_t^1, X_t^2)] &= \sum_{i,j=0}^n d(s_i, s_j) \hat{p}_i \hat{p}_j, \\ \widehat{E}[d(X_t, X_t^r)] &= \sum_{i,j=0}^n d(s_i, s_j) \hat{p}_i \hat{p}_{n-j}, \end{aligned} \quad (7)$$

respectively. Table 1 also contains some features assessing the serial dependence in an ordinal process. In this context, one of the most common quantities is the so-called ordinal Cohen's  $\kappa$ , which measures the relative deviation of the dispersion for dependent and independent random variables at a given lag  $l \in \mathbb{Z}$ . This quantity can take either positive or negative values, with its upper bound being 1 and its lower bound being dependent on the underlying distance  $d$ . A sample version of  $\kappa_d(l)$ , denoted by  $\widehat{\kappa}_d(l)$ , is obtained by using  $\widehat{\text{disp}}_d$  and the standard estimate of  $E[d(X_t, X_{t-l})]$  defined as

$$\hat{E}[d(X_t, X_{t-l})] = \frac{1}{T-l} \sum_{t=l+1}^T d(\bar{X}_t, \bar{X}_{t-l}). \quad (8)$$

A detailed analysis of the marginal quantities in Table 1 plus the ordinal Cohen's  $\kappa$  is given in [4]. In particular, the asymptotic properties of the corresponding estimates are derived and their behavior is analyzed in some simulation experiments.

**Table 1.** Some features of an ordinal stochastic process (top) and measuring serial cross-dependence between an ordinal and a numerical process (bottom). DIVC stands for diversity coefficient. TCC stands for total cumulative correlation, TMCLC stands for total mixed cumulative linear correlation, and TMCQC stands for total mixed cumulative quantile correlation.

Ordinal Measure	Definition	Range	Type
Location (standard)	$x_{\text{loc},d} = \arg \min_{x \in \mathcal{S}} E[d(X_t, x)]$	$\mathcal{S}$	Marginal
Location (with respect to $s_0$ )	$x_{\text{loc},d}^0 = \arg \min_{x \in \mathcal{S}}  E[d(X_t, s_0)] - d(x, s_0) $	$\mathcal{S}$	Marginal
Dispersion (standard)	$\text{disp}_{\text{loc},d} = E[d(X_t, x_{\text{loc},d})]$	$[0, d_0^n]$	Marginal
Dispersion (DIVC)	$\text{disp}_d = E[d(X_t^1, X_t^2)]$	$[0, d_0^n]$	Marginal
Asymmetry	$\text{asym}_d = E[d(X_t, X_t^*)] - \text{disp}_d$	$[0, d_0^n]$	Marginal
Skewness	$\text{skew}_d = E[d(X_t, s_n)] - E[d(X_t, s_0)]$	$[-d_0^n, d_0^n]$	Marginal
Ordinal Cohen's $\kappa$	$\kappa_d(l) = \frac{\text{disp}_d - E[d(X_t, X_{t-l})]}{\text{disp}_d}$	-	Serial
TCC	$\Psi^c(l) = \frac{1}{n^2} \sum_{i,j=0}^{n-1} \psi_{ij}(l)^2$	$[0, 1]$	Serial
Ordinal and numerical measure	Definition	Range	Type
TMCLC	$\Psi_1^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \psi_i^*(l)^2$	$[0, 1]$	Serial
TMCQC	$\Psi_2^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \int_0^1 \psi_i^\rho(l)^2 d\rho$	$[0, 1]$	Serial

The serial dependence of an ordinal process can be evaluated by means of alternative quantities which do not pertain to the approach based on expected distances. First, let us define the cumulative binarization of the process  $X_t$  as the multivariate process  $\{Y_t = (Y_{t,0}, \dots, Y_{t,n-1})^\top, t \in \mathbb{Z}\}$  such that  $Y_{t,i} = 1$  if  $X_t \leq s_i$ ,  $i = 0, \dots, n-1$ . By considering pairwise correlations in the cumulative binarization and fixing a lag  $l \in \mathbb{Z}$ , we obtain the quantities

$$\psi_{ij}(l) = \text{Corr}(Y_{t,i}, Y_{t-l,j}) = \frac{f_{ij}(l) - f_i f_j}{\sqrt{f_i(1-f_i)f_j(1-f_j)}}. \quad (9)$$

$i, j = 0, \dots, n-1$ . The features in (9) are very convenient because they play a similar role than the autocorrelation function of a numerical stochastic process. A measure of dependence at lag  $l$  can be obtained by considering the sum of the squares of all features  $\psi_{ij}(l)$ . In this way, we define the total cumulative correlation (TCC) as

$$\Psi^c(l) = \frac{1}{n^2} \sum_{i,j=0}^{n-1} \psi_{ij}(l)^2. \quad (10)$$

An estimate of the previous quantity can be obtained by considering  $\hat{\Psi}^c(l) = \frac{1}{n^2} \sum_{i,j=0}^{n-1} \hat{\psi}_{ij}^c(l)^2$ , where  $\hat{\psi}_{ij}^c(l)$  is the natural estimate of  $\psi_{ij}^c(l)$  obtained by replacing  $f_i, f_j$  and  $f_{ij}(l)$  by  $\hat{f}_i, \hat{f}_j$  and  $\hat{f}_{ij}(l)$  in (9) computed from the realization  $\bar{X}_t$ .

Another interesting phenomenon that can be analyzed when dealing with an ordinal process is to measure the degree of cross-dependence that the process displays with respect to a given real-valued process. To this aim, let  $\{Z_t, t \in \mathbb{Z}\}$  be a strictly stationary real-valued process with variance  $\sigma^2$  and consider the correlation

$$\psi_i^*(l) = \text{Corr}(Y_{t,i}, Z_{t-l}) = \frac{\text{Cov}(Y_{t,i}, Z_{t-l})}{\sqrt{f_i(1-f_i)\sigma^2}}, \quad (11)$$

$i = 0, \dots, n-1$ , which evaluates the level of linear dependence between state  $s_i$  of process  $X_t$  and the process  $Z_t$  at a given lag  $l \in \mathbb{Z}$ . A more complete measure assessing general types of dependence can be constructed by defining the quantity

$$\psi_i^\rho(l) = \text{Corr}(Y_{t,i}, I(Z_{t-l} \leq q_{Z_t}(\rho))) = \frac{\text{Cov}(Y_{t,i}, I(Z_{t-l} \leq q_{Z_t}(\rho)))}{\sqrt{f_i(1-f_i)\rho(1-\rho)}}, \quad (12)$$

$i = 0, \dots, n-1$ , where  $\rho \in (0, 1)$  is a probability level,  $q_{Z_t}$  denotes the quantile function of process  $Z_t$ . Note that, by considering different values for  $\rho$ , dependence at different levels at lag  $l$  can be evaluated between processes  $X_t$  and  $Z_t$ .

The features of the form  $\psi_i^*(l)$  can be combined in a proper way to obtain a suitable measure of the average linear correlation between an ordinal and a numerical process. In this way, we define the total mixed cumulative linear correlation (TMCLC) at lag  $l$  as

$$\Psi_1^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \psi_i^*(l)^2. \quad (13)$$

Analogously, a measure of the average quantile correlation between both processes, so-called the total mixed cumulative quantile correlation (TMCQC) at lag  $l$ , can be defined as

$$\Psi_2^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \int_0^1 \psi_i^\rho(l)^2 d\rho. \quad (14)$$

Note that both quantities  $\Psi_1^m(l)$  and  $\Psi_2^m(l)$  (see the lower part of Table 1) are naturally defined in the range  $[0, 1]$ , with the value 0 being reached in the case of null cross-dependence between  $X_t$  and  $Z_t$ . On the contrary, larger values indicate a stronger degree of cross-dependence between both processes.

Natural estimates of  $\Psi_1^m(l)$  and  $\Psi_2^m(l)$ , denoted by  $\hat{\Psi}_1^m(l)$  and  $\hat{\Psi}_2^m(l)$ , respectively, can be obtained by considering standard estimates of  $\psi_i^*(l)$  and  $\psi_i^\rho(l)$ , denoted by  $\hat{\psi}_i^*(l)$  and  $\hat{\psi}_i^\rho(l)$ , respectively. To compute the latter estimates, a  $T$ -length realization of the bivariate process  $\{(X_t, Z_t), t \in \mathbb{Z}\}$ , that is  $(\bar{X}_t, \bar{Z}_t) = \{(\bar{X}_1, \bar{Z}_1), \dots, (\bar{X}_T, \bar{Z}_T)\}$ , is needed. In this way, estimates  $\hat{\psi}_i^*(l)$  and  $\hat{\psi}_i^\rho(l)$  take the form

$$\begin{aligned} \hat{\psi}_i^*(l) &= \frac{\widehat{\text{Cov}}(Y_{t,i}, Z_{t-l})}{\sqrt{\hat{f}_i(1-\hat{f}_i)\hat{\sigma}^2}}, \\ \hat{\psi}_i^\rho(l) &= \frac{\widehat{\text{Cov}}(Y_{t,i}, I(Z_{t-l} \leq q_{Z_t}(\rho)))}{\sqrt{\hat{f}_i(1-\hat{f}_i)\rho(1-\rho)}}, \end{aligned} \quad (15)$$

where  $\widehat{\text{Cov}}(\cdot, \cdot)$  denotes the standard estimate of the covariance between two random variables, and  $\hat{\sigma}^2$  is the standard estimate of the variance of process  $Z_t$  computed from the realization  $\bar{Z}_t$ . Estimates in (15) give rise to the quantities  $\hat{\Psi}_1^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \hat{\psi}_i^*(l)^2$  and  $\hat{\Psi}_2^m(l) = \frac{1}{n} \sum_{i=0}^{n-1} \int_0^1 \hat{\psi}_i^\rho(l)^2 d\rho$ .



### 3. Main Functions in **otsfeatures**

This section is devoted to present the main content of package **otsfeatures**. 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.

#### 3.1. Available Datasets in **otsfeatures**

The package **otsfeatures** contains some OTS datasets which can be employed to compute ordinal features, evaluate different data mining algorithms, or simply for illustrative purposes. Specifically, **otsfeatures** includes two databases of financial time series that were introduced by [12]. In addition, three simulated data collections which were also used in [12] for the evaluation of clustering algorithms are provided. A description regarding the databases which are available in **otsfeatures** is provided below.

- Financial datasets.** The first financial dataset contains credit ratings according to Standard & Poors (S&P) for the 27 countries of the European Union (EU) plus the United Kingdom [4,12]. Each country is described by means of a monthly time series with values ranging from “D” (worst rating) to “AAA” (best rating). Specifically, the whole range consists of the  $n + 1 = 23$  states  $s_0, \dots, s_{22}$ , given by “D”, “SD”, “R”, “CC”, “CCC−”, “CCC”, “CCC+”, “B−”, “B”, “B+”, “BB−”, “BB”, “BB+”, “BBB−”, “BBB”, “BBB+”, “A−”, “A”, “A+”, “AA−”, “AA”, “AA+”, and “AAA”, respectively. The sample period spans from January 2000 to December 2017, thus resulting serial realizations of length  $T = 216$ . The second database consists of 9402 time series for Austrian men entering the labor market between 1975 and 1980 at an age of at most 25 years [8]. The time series represent gross wages categories in May of successive years, which are labeled with the integers from 0 to 5. The quintiles of the income distribution for a given year were used to define the wage categories. In this way, category 0 represents individuals with the lowest incomes, while category 5 represents individuals with the highest incomes. The series exhibit individual lengths ranging from 2 to 32 years with the median length being equal to 22. Note that, as a natural ordering exists in the set of wage categories, the corresponding time series can be naturally treated as OTS.
- Synthetic datasets.** Each one of the synthetic datasets is associated with a particular ordinal model concerning the underlying count process of a given OTS, namely binomial  $AR(p)$  [19], binomial  $INARCH(p)$  [20], and ordinal logit  $AR(1)$  (see Examples 7.4.6 and 7.4.8 in [3]) models for the first, second, and third database, respectively. In all cases, the corresponding collection contains 80 series with  $n + 1 = 6$  categories and length  $T = 600$ , which are split into 4 groups of 20 series each. All series in a given dataset were generated from the corresponding type of process but the coefficients of the generating model are different between groups. The specific coefficients were chosen by considering Scenarios 1, 2, and 3 in [12]. According to the structure of these data objects, the existence of 4 different classes can be assumed.

It is worth highlighting that the databases available in **otsfeatures** were already considered in the literature for several purposes. Specifically, the dataset of credit ratings was employed by [4] to perform data analysis of OTS, while the database of Austrian employees was used by [8] to carry out clustering of categorical time series. Additionally, in [12], both collections were considered for the application of clustering procedures specifically designed to deal with OTS. Thus, it is clearly beneficial for the user to have available the corresponding databases through **otsfeatures**. On the other hand, we should note that, in each one of the synthetic datasets, the different classes can be distinguished by means of both marginal distributions and serial dependence patterns. Hence, these data objects are suitable to evaluate the effectiveness of the features in Table 1 for several machine learning problems. In fact, the usefulness of these features to carry out clustering and classification tasks (among others) in these databases is illustrated in Section 4.3. Table 2 contains a summary of the 5 datasets included in **otsfeatures**. Specifically, the last column

contains the number of classes existing in a given data collection according to a context of supervised classification. For instance, 4 different classes are assumed to exist in the synthetic databases due to the fact that the 80 time series in each one were generated from 4 different stochastic processes. It is worth highlighting that datasets **CreditRatings** and **AustrianWages** do not contain clearly defined classes, and thus the notation “-” was used in the last column for these databases.

**Table 2.** Summary of the datasets included in **otsfeatures**. The notation No. Series,  $T$ ,  $|\mathcal{S}|$ , and No. Classes stands for the number of series, the length of the series, the number of categories in the range of the series and the number of classes existing in a given dataset.

Dataset	Object	No. Series	$T$	$ \mathcal{S} $	No. Classes
Credit Ratings	<i>CreditRatings</i>	28	216	23	-
Austrian Wages	<i>AustrianWages</i>	9402	Variable	6	-
Synthetic I	<i>SyntheticData1</i>	80	600	6	4
Synthetic II	<i>SyntheticData2</i>	80	600	6	4
Synthetic III	<i>SyntheticData3</i>	80	600	6	4

### 3.2. Functions for Inferential Tasks

In this section, we present some of the tools available in **otsfeatures** to perform classical statistical tasks. In particular, we first describe one specific plot which can be used to analyze the serial dependence structure of a given ordinal series. Afterwards, we give an overview of some functions allowing to carry out hypothesis testing and the construction of confidence intervals for the quantities introduced in Section 2.

#### 3.2.1. Serial Dependence Plot

When analyzing a real-valued processes, the autocorrelation function is a classical tool for describing the corresponding serial dependence structure. Note that, in the ordinal setting, this function can still be employed by considering the underlying count process  $C_t$  introduced in Section 2, which is indeed real-valued. However, using the autocorrelation function in this context has several drawbacks, since one is treating the ordinal process as a numerical process, thus ignoring the available information about the dissimilarity between the different ordinal categories. Therefore, an alternative, more suitable tool is required to examine the serial dependence patterns of an ordinal process. In this regard, one interesting possibility consists of considering the quantity  $\kappa_d(l)$  in order to evaluate the degree of dependence exhibited by the process at a given lag  $l \in \mathbb{Z}$ . In fact, this quantity takes the value of 0 for an i.i.d. process, while positive or negative values are associated with different types of dependence structures. Clearly, in practice, one often works with the  $T$ -length realization  $\bar{X}_t$  and computes the estimated Cohen’s  $\kappa$ ,  $\hat{\kappa}_d(l)$ , which can be used to describe the serial dependence patterns of the underlying ordinal process.

It is worth highlighting that the asymptotic distribution of the previous estimate is well-known in the particular case of the distance  $d$  being the block distance. Specifically, according to Theorem 7.2.1 in [4], the distribution of the estimate  $\hat{\kappa}_{d_{0,1}}(l)$  can be approximated by a normal distribution with mean  $-\frac{1}{T}$  and variance  $\frac{4}{T \text{disp}_{d_{0,1}}^2} \sum_{k,l=0}^{n-1} (\hat{f}_{\min\{k,l\}} - \hat{f}_k \hat{f}_l)^2$ .

The previous asymptotic result is rather useful in practice, since it can be used to test the null hypothesis of serial independence at lag  $l$ . In particular, critical values for a given significance level  $\alpha$  can be computed, and these quantities do not depend on the specific lag. Thus, a serial dependence graph analogous to the ACF-based plot in the real-valued case can be constructed. Specifically, after setting a maximum lag of interest,  $L$ , the values of  $\hat{\kappa}_{d_{0,1}}(l)$  for lags ranging from 1 to  $L$  are simultaneously depicted in one graph. Next, the corresponding critical values are added to the plot by means of a horizontal lines.

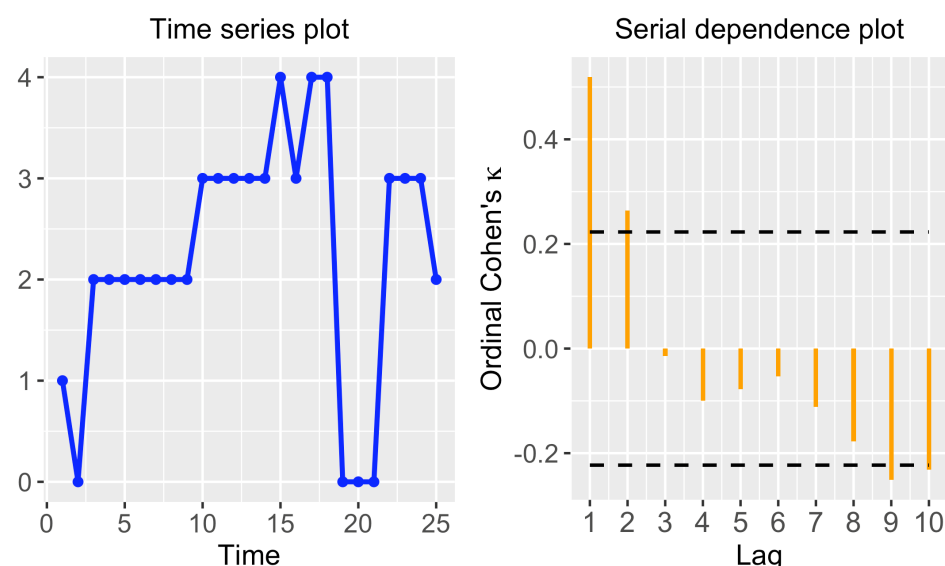


According to the asymptotic approximation for  $\hat{\kappa}_{d_{0,1}}(l)$ , the critical values for an arbitrary significance level  $\alpha$  are given by

$$\pm \frac{2\sqrt{\sum_{k,l=0}^{n-1} (\hat{f}_{\min\{k,l\}} - \hat{f}_k \hat{f}_l)^2} z_{1-\alpha/2}}{\sqrt{T} \widehat{\text{disp}}_{d_{0,1}}} - \frac{1}{T}, \quad (16)$$

where  $z_\tau$  denotes the  $\tau$ -quantile of the standard normal distribution. The corresponding graph allows one to easily identify the collection of significant lags for a given ordinal series. Similarly to the autocorrelation plot in the numerical setting, serial dependence plots for OTS can be used for several purposes, including model selection or identification of regular patterns in the series among others.

The right panel of Figure 1 shows the serial dependence plot based on  $\hat{\kappa}_{d_{0,1}}(l)$  for one of the time series in the dataset *AustrianWages*. A maximum lag  $L = 10$  was considered. The function `plot_cohens_kappa()` was employed to construct the graph. It is worth remarking that, if the argument `plot = FALSE` is used in this function, then the output is not the serial dependence plot but a list containing the corresponding  $p$ -values and critical values.



**Figure 1.** Time series plot (left panel) and serial dependence plot based on  $\hat{\kappa}_{d_{0,1}}(l)$  (right panel) for one of the series in dataset *AustrianWages*. The dashed lines indicate the critical values regarding the null hypothesis of the corresponding quantity being zero.

### 3.2.2. Hypothesis Testing and Confidence Intervals

The package **otsfeatures** allows us to perform hypothesis tests for alternative quantities in addition to  $\kappa_{d_{0,1}}(l)$ . In particular, there are some functions for testing that the quantities  $\text{disp}_{d_{0,1}}$ ,  $\text{asym}_{d_{0,1}}$  and  $\text{skew}_{d_{0,1}}$  are equal to some specified values employing the corresponding estimates. In addition, confidence intervals for these quantities can be constructed through some commands available in the package. In both cases, the corresponding implementations rely on the asymptotic results provided in Theorem 7.1.1 in [4]. It is worth highlighting that these results are valid for the general case in which dependence between observations exist. However, when dealing with i.i.d. data, the corresponding expressions for the asymptotic means and variances are still valid but they are simplified. In this regard, package **otsfeatures** gives the user the possibility of performing hypothesis tests and constructing confidence intervals for i.i.d. data (see Theorem 4.1 in [4]). This is indicated to the corresponding functions by using the argument `temporal = FALSE`.

A summary of the main functions in **otsfeatures** allowing one to perform inferential tasks is given in Table 3.

**Table 3.** Some functions for inference tasks implemented in **otsfeatures**.

Output	Function in Otsfeatures
Serial dependence plot for $\widehat{\kappa}_{d_{o,1}}(l)$	<code>plot_ordinal_cohens_kappa()</code>
Test based on $\widehat{\kappa}_{d_{o,1}}(l)$	<code>plot_ordinal_cohens_kappa(plot = FALSE)</code>
Test based on $\widehat{\text{disp}}_{d_{o,1}}$	<code>test_ordinal_dispersion()</code>
Test based on $\widehat{\text{asym}}_{d_{o,1}}$	<code>test_ordinal_asymmetry()</code>
Test based on $\widehat{\text{skew}}_{d_{o,1}}$	<code>test_ordinal_skewness()</code>
Confidence interval for $\widehat{\text{disp}}_{d_{o,1}}$	<code>ci_ordinal_dispersion()</code>
Confidence interval for $\widehat{\text{asym}}_{d_{o,1}}$	<code>ci_ordinal_asymmetry()</code>
Confidence interval for $\widehat{\text{skew}}_{d_{o,1}}$	<code>ci_ordinal_skewness()</code>

### 3.3. Functions for Feature Extraction in **otsfeatures**

The package **otsfeatures** contains several functions allowing one to compute well-known statistical quantities for OTS measuring either marginal or serial properties. All commands of this type are based on the estimated features presented in Section 2. A summary of the corresponding functions is given in Table 4. In Section 4.3, the use of several functions for feature extraction available in **otsfeatures** is illustrated through several examples.

**Table 4.** Some functions for feature extraction implemented in **otsfeatures**.

Features	Function in Otsfeatures	Features	Function in Otsfeatures
$(\widehat{p}_0, \dots, \widehat{p}_n)$	<code>marginal_probabilities()</code>	$\widehat{\text{disp}}_d$	<code>ordinal_dispersion_2()</code>
$(\widehat{p}_{ij}(l))_{0 \leq i,j \leq n}$	<code>joint_probabilities()</code>	$\widehat{\text{asym}}_d$	<code>ordinal_asymmetry()</code>
$(\widehat{f}_0, \dots, \widehat{f}_{n-1})$	<code>c_marginal_probabilities()</code>	$\widehat{\text{skew}}_d$	<code>ordinal_skewness()</code>
$(\widehat{f}_{ij}(l))_{0 \leq i,j \leq n-1}$	<code>c_joint_probabilities()</code>	$\widehat{\kappa}_d(l)$	<code>ordinal_cohens_kappa()</code>
$\widehat{x}_{\text{loc},d}$	<code>ordinal_location_1()</code>	$\widehat{\Psi}^c(l)$	<code>total_c_cor()</code>
$\widehat{x}_{\text{loc},d}^0$	<code>ordinal_location_2()</code>	$\widehat{\Psi}_1^m(l)$	<code>total_mixed_c_cor()</code>
$\widehat{\text{disp}}_{\text{loc},d}$	<code>ordinal_dispersion_1()</code>	$\widehat{\Psi}_2^m(l)$	<code>total_mixed_c_qcor()</code>

## 4. Using the **otsfeatures** Package—An Illustration

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

### 4.1. Some Generalities about **otsfeatures**

In **otsfeatures**, a  $T$ -length OTS with range  $\mathcal{S} = \{s_0, s_1, \dots, s_n\}$ ,  $\bar{X}_t = \{\bar{X}_1, \dots, \bar{X}_T\}$ , is defined through a vector of length  $T$  whose possible values are the integer numbers from 0 to  $n$ . More precisely, the realization  $\bar{X}_t$  is represented by using the associated realization of the generating count process  $C_t$ , that is,  $\bar{C}_t = \{\bar{C}_1, \dots, \bar{C}_T\}$  such that  $\bar{X}_j = s_{\bar{C}_j}$ ,  $j = 1, \dots, T$ . Note that the main advantage of this approach relies on the fact that only numerical vectors are needed for the representation of ordinal series.

The majority of functions in the package take as input a single OTS. For instance, functions in Table 4 return by default the corresponding estimate. Some of these functions admit the argument `features = TRUE`. In that case, the function returns a vector which contains the individual quantities which are considered to construct the corresponding estimate. For instance, the function `total_c_cor()` computes by default the estimate  $\widehat{\Psi}^c(l)$ . However, if we employ the argument `features = TRUE`, a matrix whose  $(i, j)$  entry contains the quantity  $\widehat{\psi}_{ij}(l)$  is returned. In fact, the extraction of the individual components of some estimates can be very useful for several purposes. Functions `ots_plot()` and `plot_ordinal_cohens_kappa()` with the default settings produce the corresponding time series

plot and serial dependence graph, respectively. On the contrary, the remaining functions and function `plot_ordinal_cohens_kappa()` with `plot = FALSE` return the results of the corresponding hypothesis tests, namely the test statistic, the critical value for a given significance level used as input, and the  $p$ -value. It is worth remarking that most commands in **otsfeatures** require the corresponding states to be specified in a vector of the form  $(0, 1, \dots, n)$ . This is done by means of the argument `states`. In this way, several issues can be avoided. For instance, a particular realization may not include all the underlying ordinal values. Therefore, when analyzing such a series, one could ignore the existence of some states. This is properly solved by using the argument `states`.

The databases included in **otsfeatures** are defined by the means of a list named as indicated in the first column of Table 2. In the case of the synthetic databases, each list contains two elements, which are described below.

- The element called *data* is a list of vectors with the ordinal series of the corresponding collection.
- The element named *classes* includes a vector of class labels associated with the objects in *data*.

On the other hand, the lists associated with datasets *CreditRatings* and *AustrianWages* only include the element *data*, as there are no underlying class labels for these data collections.

Let us take a look at one time series in dataset *AustrianWages*, which represents a specific employee of the Austrian labor market.

```
> library(otsfeatures)
> AustrianWages$data[[10]]
[1] 3 3 3 3 3 0 0 3 2 0 4 0 0 3 3 3 4 5 4 4 4 5
```

In this series, the corresponding wage categories are identified with the integers from 0 to 5 as explained in Section 3.1 (category 0 represents the lowest incomes and category 5, the highest incomes). In this way, the previous sequence represents an individual who started with a moderate wage (category 3), then decreased his income level and finally ended up in the highest wage category. Note that this representation of the series by means of integer numbers provides a simple way of quickly examining the corresponding ordinal values.

#### 4.2. Performing Inferential Tasks

The functions described in Section 3.2 allow the user to obtain valuable information from a given ordinal series. Let us start by analyzing one of the time series in the dataset *AustrianWages*. Before carrying out inferential tasks, we are going to visualize the corresponding time series as a preliminary step. To this aim, we can employ the function `ots_plot()`, which takes as input the time series we want to represent and a vector containing the different states.

```
ots_plot(AustrianWages$data[[100]], states = 0 : 5,
labels = 0 : 5)
```

We also employed the argument `labels = 0:5` to indicate that the states  $s_0$  to  $s_5$  are labeled with the integers from 0 to 5, since this is the labeling used in the original dataset (see Section 3.1). The corresponding graph is provided in the left panel of Figure 1. This series corresponds to an individual who belonged to all income levels except for the highest one (5). It is worth highlighting that, as the different states are located in the  $y$ -axis in increasing order, the plot is rather intuitive. In addition, note that, for the sake of simplicity, the different categories are treated as equidistant. Specifically, the graph is constructed by considering the block distance  $d_{0,1}$  between states, which is not always suitable, since the true underlying distance often depends on the specific context. Therefore, the graph produced by function `ots_plot()` should not be treated as an accurate plot of the corresponding OTS, but as a rough representation thereof.

As stated in Section 3.2.1, function `plot_ordinal_cohens_kappa()` in **otsfeatures** allows to construct a serial dependence plot based on the estimate  $\hat{\kappa}_{d_{0,1}}(l)$ . Let us represent such plot for the series in the left panel of Figure 1.

```
> sd_plot <- plot_ordinal_cohens_kappa(series = AustrianWages$data[[100]],
states = 0 : 5)
```

By default, the function considers lags from 1 to 10 (argument *max\_lag*) and a significance level  $\alpha = 0.05$  for the corresponding test (argument  $\alpha$ ). The resulting graph is given in the right panel of Figure 1. As the standard autocorrelation plot, the corresponding estimates are displayed in a sequential order, with dashed lines indicating the critical values for the associated test. In this case, the serial dependence plot indicates significant dependence at lags 1 and 2. Moreover, dependence at lags 9 and 10 could also be considered significant, but this may be due to chance, since multiple tests are simultaneously carried out. In addition to the dependence plot, function *plot\_ordinal\_cohens\_kappa()* also produces numerical outputs. For instance, the corresponding *p*-values can be obtained by using the argument *plot = FALSE*.

```
> sd_plot <- plot_ordinal_cohens_kappa(series = AustrianWages$data[[100]],
states = 0 : 5, plot = FALSE)
> round(sd_plot$p_values, 2)
[1] 0.00 0.02 0.68 0.30 0.38 0.49 0.26 0.11 0.03 0.04
```

The *p*-values in the previous output corroborate that the quantity  $\kappa_{d_{0,1}}(1)$  is significantly non-null, thus confirming the existence of serial dependence at the first lag. Note that the *p*-values associated with lags 2, 9 and 10 also indicate rejection of the null hypothesis at level  $\alpha = 0.05$ . However, the 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  $\alpha$ -level, could be applied to the *p*-values by executing the following command.

```
> p.adjust(round(sd_plot$p_values, 2), method = 'holm')
[1] 0.00 0.18 1.00 1.00 1.00 1.00 1.00 0.66 0.24 0.28
```

According to the corrected *p*-values, significant serial dependence still exists at lag 1, but the null hypothesis of serial independence at lags 2, 9 and 10 cannot be now rejected.

In addition to analyzing serial dependence, hypothesis tests and confidence intervals for classical ordinal quantities can be constructed by using **otsfeatures** (see Section 3.2.2). To illustrate these tasks, we consider again the previous OTS and start by testing the null hypothesis stating that the quantity  $\text{skew}_{d_{0,1}}$  is equal to 0. To this aim, we employ the function *test\_ordinal\_skewness()*, whose main arguments are the corresponding ordinal series (argument *series*) and the assumed value for  $\text{skew}_{d_{0,1}}$  (argument *true\_skewness*), which is set to zero in this example.

```
> test_os <- test_ordinal_skewness(series = AustrianWages$data[[100]],
states = 0 : 5, true_skewness = 0)
> test_os$p_value
[1] 0.4239951
```

The *p*-value of the test resulted as 0.424. Therefore, the null hypothesis cannot be rejected at any reasonable significance level and we can assume that the series was generated from an ordinal process with 0 skewness. For illustrative purposes, let us repeat the previous test by setting *true\_skewness* = 2.

```
> test_os <- test_ordinal_skewness(series = AustrianWages$data[[100]],
states = 0 : 5, true_skewness = 2)
> test_os$p_value
[1] 0.02287435
```

This time, the *p*-value indicates that the null hypothesis should be rejected at the standard significance level  $\alpha = 0.05$ ; that is, we could assume that the true skewness is

different from 2 at that level. However, this is no longer the case for stricter significance levels (e.g.,  $\alpha = 0.01$ ).

The construction of a confidence interval for the quantity  $\text{skew}_{d_{o,1}}$  can be easily performed by using the function `ci_ordinal_skewness()`. By default, a confidence level of 0.95 is considered (argument `level`).

```
> ci_os <- ci_ordinal_skewness(series = AustrianWages$data[[100]],
states = 0 : 5)
> ci_os
Lower bound Upper bound
1 -0.7547583 1.794758
```

The lower and upper bounds of the confidence interval are given by  $-0.75$  and  $1.79$ , respectively. It is worth remarking that, as we are dealing with a rather short time series ( $T = 25$ ), the interval is quite broad. In addition, note that 0 is included in the interval, which is coherent with the results of the first hypothesis test for  $\text{skew}_{d_{o,1}}$  above. Let us now construct a confidence interval by considering a less strict confidence level, namely 0.90.

```
> ci_os <- ci_ordinal_skewness(series = AustrianWages$data[[100]],
states = 0 : 5, level = 0.90)
> ci_os
Lower bound Upper bound
1 -0.5498109 1.589811
```

As expected, the new interval has a shorter length than the previous one.

Inferential tasks for quantities  $\text{disp}_{d_{o,1}}$ , and  $\text{asym}_{d_{o,1}}$  can be carried out in an analogous way by using the corresponding functions (see Table 3). Moreover, these commands can also be used when dealing with i.i.d. data by using the argument `temporal = FALSE`.

#### 4.3. Performing Data Mining Tasks

Jointly used with external functions, **otsfeatures** becomes a versatile and helpful tool to carry out different data-mining tasks involving ordinal series. In this section, for illustrative purposes, we focus our attention on three important problems, namely classification, clustering, and outlier detection.

##### 4.3.1. Performing OTS 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 *SyntheticData1*, which contains 80 series generated from 4 different stochastic processes, each one of them giving rise to 20 OTS. The underlying processes are given by two binomial AR(1) and two binomial AR(2) models. Thus, each series in the dataset *SyntheticData1* has an associated class label determined by the corresponding generating process. Using the necessary functions, each series is replaced by a feature vector given by the quantities  $\widehat{x}_{\text{loc},d_{o,1}}$ ,  $\widehat{\text{disp}}_{d_{o,1}}$ ,  $\widehat{\text{asym}}_{d_{o,1}}$ ,  $\widehat{\text{skew}}_{d_{o,1}}$ ,  $\widehat{\kappa}_{d_{o,1}}(1)$  and  $\widehat{\kappa}_{d_{o,1}}(2)$ . In all cases, the argument `distance = 'Block'` (default) is used to indicate that the block distance should be employed as the underlying block distance between states.

```
> features_1 <- unlist(lapply(SyntheticData1$data,
ordinal_location_1, states = 0 : 5, distance = 'Block'))
> features_2 <- unlist(lapply(SyntheticData1$data,
ordinal_dispersion_2, states = 0 : 5, distance = 'Block'))
> features_3 <- unlist(lapply(SyntheticData1$data,
ordinal_asymmetry, states = 0 : 5, distance = 'Block'))
> features_4 <- unlist(lapply(SyntheticData1$data,
ordinal_skewness, states = 0 : 5, distance = 'Block'))
```

```
> features_5 <- unlist(lapply(SyntheticData1$data,
  ordinal_cohens_kappa, states = 0 : 5, distance = 'Block', lag = 1))
> features_6 <- unlist(lapply(SyntheticData1$data,
  ordinal_cohens_kappa, states = 0 : 5, distance = 'Block', lag = 2))
> feature_dataset <- cbind(features_1, features_2, features_3,
  features_4, features_5, features_6)
```

Note that the  $i$ th row of the object *feature\_dataset* contains estimated values characterizing the marginal and serial behavior of the  $i$ th OTS in the dataset. Therefore, several standard classification algorithms can be applied to these matrix by means of the R package **caret** [21]. 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,
  SyntheticData1$classes))
> colnames(df_feature_dataset)[7] <- 'Class'
> df_feature_dataset[,7] <- factor(df_feature_dataset[,7])
```

The function *train()* allows one 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 neighbours ( $k$ NN) 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')
> 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] 0.95
```

The  $k$ NN classifier achieves an accuracy of 0.95 in the dataset *SyntheticData1*. Specifically, it produces only 4 misclassifications. Next, we study the performance of the random forest and the linear discriminant analysis. To this aim, we need to set *method = 'rf'* and *method = 'lda'*, respectively.

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

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

Both approaches reach a perfect accuracy of 1, thus improving the predictive effectiveness of the  $k$ NN classifier. For illustrative purposes, let us analyze the performance of the previous classifiers when the Hamming distance between ordinal categories is taken into account, which is indicated through the argument *distance = 'Hamming'*.



```

> features_1 <- unlist(lapply(SyntheticData1$data,
ordinal_location_1, states = 0 : 5, distance = 'Hamming'))
> features_2 <- unlist(lapply(SyntheticData1$data,
ordinal_dispersion_2, states = 0 : 5, distance = 'Hamming'))
> features_3 <- unlist(lapply(SyntheticData1$data,
ordinal_asymmetry, states = 0 : 5, distance = 'Hamming'))
> features_4 <- unlist(lapply(SyntheticData1$data,
ordinal_skewness, states = 0 : 5, distance = 'Hamming'))
> features_5 <- unlist(lapply(SyntheticData1$data,
ordinal_cohens_kappa, states = 0 : 5, distance = 'Hamming', lag = 1))
> features_6 <- unlist(lapply(SyntheticData1$data,
ordinal_cohens_kappa, states = 0 : 5, distance = 'Hamming', lag = 2))
> feature_dataset <- cbind(features_1, features_2, features_3,
features_4, features_5, features_6)

> df_feature_dataset <- data.frame(cbind(feature_dataset,
SyntheticData1$classes))
> colnames(df_feature_dataset)[7] <- 'Class'
> df_feature_dataset[,7] <- factor(df_feature_dataset[,7])

> model_knn <- train(Class~., data = df_feature_dataset,
trControl = train_control, method = 'knn')
> max(model_knn$results$Accuracy)
[1] 0.975

> model_rf <- train(Class~., data = df_feature_dataset,
trControl = train_control, method = 'rf')
> max(model_rf$results$Accuracy)
[1] 1

> model_lda <- train(Class~., data = df_feature_dataset,
trControl = train_control, method = 'lda')
> max(model_lda$results$Accuracy)
[1] 0.975

```

By considering the distance  $d_H$ , the  $k$ NN classifier slightly improves its performance while the linear discriminant analysis shows a small decrease in predictive effectiveness. The random forest still reaches perfect results. The classification ability of alternative sets of features, as well as the behavior of any other classifier, can be examined in an analogous way as above.

#### 4.3.2. Performing OTS Clustering

The package **otsfeatures** also provides an excellent framework to carry out clustering of ordinal sequences. Let us consider now the dataset *SyntheticData2* 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 (a specific binomial INARCH( $p$ ) 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 two-dimensional space preserving the original values as much as possible (by minimizing a loss function). For instance, we are going to construct the 2DS for dataset *SyntheticData2* by using two specific distances between CTS proposed by [12] and denoted by  $\hat{d}_1$  and  $\hat{d}_{PMF}$ . More specifically, given two OTS  $\bar{X}_t^{(1)}$  and  $\bar{X}_t^{(2)}$ , the metrics  $\hat{d}_1$  and  $\hat{d}_{PMF}$  are defined as follows:

$$\begin{aligned}\hat{d}_1(\bar{X}_t^{(1)}, \bar{X}_t^{(2)}) &= \sum_{i=0}^{n-1} \left( \hat{f}_i^{(1)} - \hat{f}_i^{(2)} \right)^2 + \sum_{k=1}^L \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \left( \hat{f}_{ij}^{(1)}(l_k) - \hat{f}_{ij}^{(2)}(l_k) \right)^2, \\ \hat{d}_{PMF}(\bar{X}_t^{(1)}, \bar{X}_t^{(2)}) &= \sum_{i=0}^n \left( \hat{p}_i^{(1)} - \hat{p}_i^{(2)} \right)^2 + \sum_{k=1}^L \sum_{i=0}^n \sum_{j=0}^n \left( \hat{p}_{ij}^{(1)}(l_k) - \hat{p}_{ij}^{(2)}(l_k) \right)^2,\end{aligned}\quad (17)$$

where  $\mathcal{L} = \{l_1, \dots, l_L\}$  is a set of  $L$  lags which must be determined in advance and the superscripts (1) and (2) indicate that the corresponding estimates are based on the realizations  $\bar{X}_t^{(1)}$  and  $\bar{X}_t^{(2)}$ , respectively. Both dissimilarities assess discrepancies between the marginal distributions (first terms) and the serial dependence structures (last terms) of both series. Therefore, they seem appropriate to group the CTS of a given collection in terms of underlying stochastic processes. However, note that the distance  $\hat{d}_1$  is based on cumulative probabilities, thus taking into account the underlying ordering existing in the series range.

Let us first create the datasets *dataset\_1* and *dataset\_2* with the features required to compute  $\hat{d}_1$  and  $\hat{d}_{PMF}$ , respectively. As the series in *SyntheticData2* were generated from binomial INARCH(1) and binomial INARCH(2) processes, we consider only the first two lags to construct the distance, i.e., we set  $\mathcal{L} = \{1, 2\}$ . We have to use the argument *features = TRUE* in the corresponding functions.

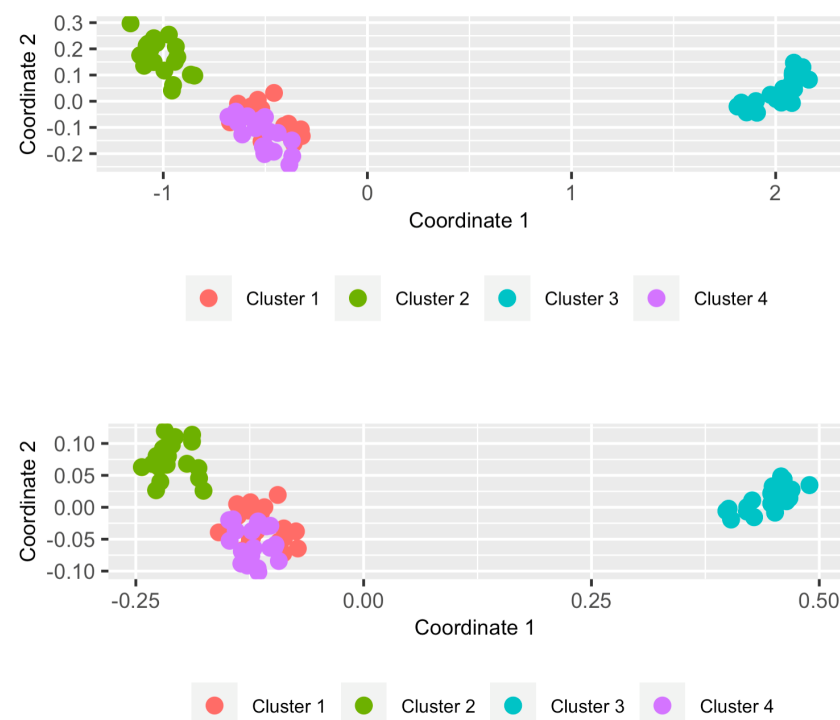
```
> list_marginal_1 <- lapply(SyntheticData2$data,
  c_marginal_probabilities, states = 0 : 5)
> list_serial_1_1 <- lapply(SyntheticData2$data,
  c_joint_probabilities, states = 0 : 5, lag = 1)
> list_serial_1_2 <- lapply(SyntheticData2$data,
  c_joint_probabilities, states = 0 : 5, lag = 2)
> dataset_marginal_1 <- matrix(unlist(list_marginal_1),
  nrow = 80, byrow = T)
> dataset_serial_1_1 <- matrix(unlist(list_serial_1_1),
  nrow = 80, byrow = T)
> dataset_serial_1_2 <- matrix(unlist(list_serial_1_2),
  nrow = 80, byrow = T)
> dataset_1 <- cbind(dataset_marginal_1, dataset_serial_1_1,
  dataset_serial_1_2)

> list_marginal_2 <- lapply(SyntheticData2$data,
  marginal_probabilities, states = 0 : 5)
> list_serial_2_1 <- lapply(SyntheticData2$data,
  joint_probabilities, states = 0 : 5, lag = 1)
> list_serial_2_2 <- lapply(SyntheticData2$data,
  joint_probabilities, states = 0 : 5, lag = 2)
> dataset_marginal_2 <- matrix(unlist(list_marginal_2),
  nrow = 80, byrow = T)
> dataset_serial_2_1 <- matrix(unlist(list_serial_2_1),
  nrow = 80, byrow = T)
> dataset_serial_2_2 <- matrix(unlist(list_serial_2_2),
  nrow = 80, byrow = T)
> dataset_2 <- cbind(dataset_marginal_2, dataset_serial_2_1,
  dataset_serial_2_2)
```

The 2DS planes can be built using the function `plot_2d_scaling()` of the R package **mlmts** [22], which takes as input a pairwise dissimilarity matrix.

```
> library(mlmts)
> distance_matrix_1 <- dist(dataset_1)
> plot_1 <- plot_2d_scaling(distance_matrix_1,
  cluster_labels = otsfeatures::SyntheticData2$classes)$plot
> distance_matrix_2 <- dist(dataset_2)
> plot_2 <- plot_2d_scaling(distance_matrix_2,
  cluster_labels = otsfeatures::SyntheticData2$classes)$plot
```

In the above code, the syntax `otsfeatures::` was employed because package **mlmts** includes a data collection which is also called *SyntheticData2*. The resulting plots are shown in Figure 2. In both cases, the points were colored according to the true partition defined by the generating models. For this, 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 OTS (the ones represented by red and purple points) exhibiting a certain degree of overlap in both plots, which suggests a high level of similarity between the corresponding generating processes.



**Figure 2.** Two-dimensional scaling planes based on distances  $\hat{d}_1$  (top panel) and  $\hat{d}_{PMF}$  (bottom panel) for the 80 series in the dataset *SyntheticData2*.

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** [23]. This function needs the pairwise distance matrix and the number of clusters. The latter argument is set to 4, since the series in dataset *SyntheticData2* were generated from 4 different stochastic processes.

```
> library(cluster)
> clustering_pam_1 <- pam(distance_matrix_1, k = 4)$clustering
> clustering_pam_2 <- pam(distance_matrix_2, k = 4)$clustering
```

The vectors *clustering\_pam\_1* and *clustering\_pam\_2* provide the respective clustering solutions based on both metrics. The evaluation of the quality of both partitions requires measuring their degree of agreement with the ground truth, which can be performed by using the Adjusted Rand Index (ARI) [24]. This index can be easily computed by means of the function *external\_validation()* of package **ClusterR** [25].

```
> library(ClusterR)
> external_validation(clustering_pam_1,
  otsfeatures::SyntheticData2$classes)
[1] 0.6545303
> external_validation(clustering_pam_2,
  otsfeatures::SyntheticData2$classes)
[1] 0.6535088
```

The ARI index is bounded between  $-1$  and  $1$  and admits a simple interpretation: the closer it is to  $1$ , the better the agreement between the ground truth and the experimental solution is. 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  $\hat{d}_1$  and  $\hat{d}_{PMF}$ , respectively attain moderate scores in this dataset when used with the PAM algorithm. In particular, both partitions are substantially similar. Note that a nonperfect value of ARI index was already expected from the 2DS plots in Figure 2 due to the overlapping character of Clusters 1 and 4.

The classical *K*-means clustering algorithm can be also executed by using **otsfeatures** utilities. In this case, we need to employ a dataset of features along with the *kmeans()* function of package **stats** [13].

```
set.seed(123)
clustering_kmeans_1 <- kmeans(dataset_1, c = 4)$cluster
external_validation(clustering_kmeans_1,
  otsfeatures::SyntheticData2$classes)
[1] 0.6545303
> set.seed(123)
> clustering_kmeans_2 <- kmeans(dataset_2, c = 4)$cluster
> external_validation(clustering_kmeans_2,
  otsfeatures::SyntheticData2$classes)
[1] 0.7237974
```

In the previous example, slightly better results are obtained when the  $\hat{d}_{PMF}$  is employed along with the *K*-means algorithm. Concerning  $\hat{d}_1$ , its clustering accuracy is exactly the same as the one associated with the PAM algorithm. The performance of alternative dissimilarities or collections of features regarding a proper identification of the underlying clustering structure could be determined by following the same steps than in the previous experiments.

#### 4.3.3. Performing Outlier Detection in OTS Datasets

The topic of outlier detection has received a lot of attention in the literature, either in the nontemporal setting (see, e.g., [26] for a review on outlier detection methods for univariate data) or in the context of time series data (see, e.g., [27] for a review on anomaly detection in time series data). Concerning the latter subject, it is worth noting that different notions of outlier are considered in this context (additive outliers, innovative outliers, and others). Here, we consider the outlying elements to be whole OTS objects. More specifically, an anomalous OTS 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 **otsfeatures** can be useful to carry out outlier identification, we create a dataset which includes two atypical elements. For it, we consider all the series in *SyntheticData3* along with the first two series in dataset *SyntheticData2*.

```
> data_outliers <- c(SyntheticData3$data, SyntheticData2$data[1:2])
```

The resulting data collection, *data\_outliers*, contains 82 OTS. The first 80 OTS 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 an ordinal logit AR(1) 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 OTS in *data\_outliers* can be identified by using the distance  $\hat{d}_1$  given in (17). First, we construct the pairwise dissimilarity matrix based on this metric for the new dataset.

```
> list_outl_1 <- lapply(data_outliers, c_marginal_probabilities,
  states = 0 : 5)
> list_outl_2 <- lapply(data_outliers, c_joint_probabilities,
  states = 0 : 5, lag = 1)
> list_outl_3 <- lapply(data_outliers, c_joint_probabilities,
  states = 0 : 5, lag = 2)
> dataset_outl_1 <- matrix(unlist(list_outl_1), nrow = 82, byrow = T)
> dataset_outl_2 <- matrix(unlist(list_outl_2), nrow = 82, byrow = T)
> dataset_outl_3 <- matrix(unlist(list_outl_3), nrow = 82, byrow = T)
> dataset_outl <- cbind(dataset_outl_1, dataset_outl_2, dataset_outl_3)
> distance_matrix_outl <- dist(dataset_outl)
```

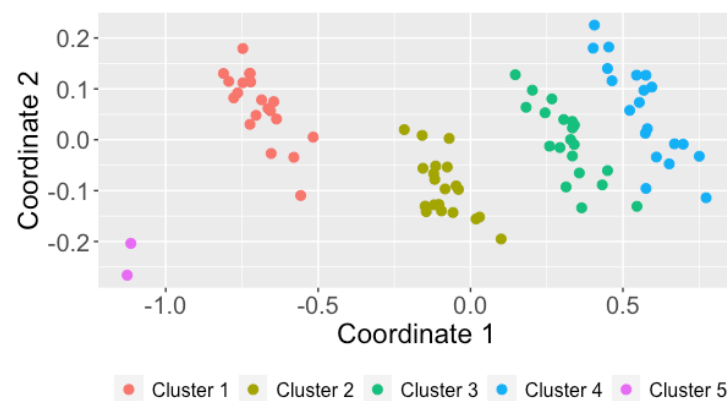
Then, we apply the mentioned two-step procedure to matrix *distance\_matrix\_outl* by running

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

The previous output corroborates that  $\hat{d}_1$  is able to properly identify the two series generated from anomalous stochastic processes. As an illustrative exercise, let us represent the corresponding 2DS plot for the dataset containing the two outlying OTS by using a different color for these elements.

```
> library(mlmts)
> labels <- c(otsfeatures::SyntheticData2$classes, 5, 5)
> plot_2d_scaling(distance_matrix_outl, cluster_labels = labels)$plot
```

The corresponding graph is shown in Figure 3. The 2DS configuration contains four groups of points which are rather well separated, plus two isolated elements representing the anomalous series appearing on the 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.



**Figure 3.** Two-dimensional scaling plane based on distance  $\hat{d}_1$  for the dataset containing 2 anomalous series.

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 OTS databases, 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., [28,29]), i.e., those elements with an extremely large scoring could be identified as outliers. In order to illustrate this approach, we consider the dataset *CreditRatings* and compute the pairwise distance matrix according to distance  $\hat{d}_{PMF}$ .

```
> list_cr_1 <- lapply(CreditRatings$data,
  marginal_probabilities, states = 0 : 22)
> list_cr_2 <- lapply(CreditRatings$data,
  joint_probabilities, states = 0 : 22, lag = 1)
> list_cr_3 <- lapply(CreditRatings$data,
  joint_probabilities, states = 0 : 22, lag = 2)
> dataset_cr_1 <- matrix(unlist(list_cr_1),
  nrow = 28, byrow = T)
> dataset_cr_2 <- matrix(unlist(list_cr_2),
  nrow = 28, byrow = T)
> dataset_cr_3 <- matrix(unlist(list_cr_3),
  nrow = 28, byrow = T)
> dataset_cr <- cbind(dataset_cr_1, dataset_cr_2,
  dataset_cr_3)
> distance_matrix_cr <- dist(dataset_cr)
```

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

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

The vector *outlier\_score* contains the sum of the distances for each of the 28 countries. Since the *i*th element of this vector can be seen as a measure of the outlying character of the *i*th series, those countries 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 Figure 4 and suggests the existence of one series with an abnormally high outlying score. Hence, this country could be considered to be anomalous and its individual properties be carefully investigated. Specifically, the



uppermost point in Figure 4 corresponds to Belgium. 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.

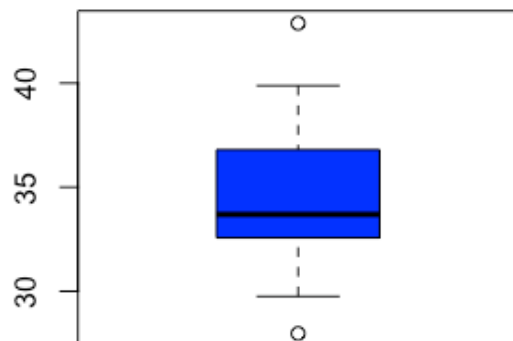


Figure 4. Boxplot of the outlying scores in dataset *CreditRatings* based on distance  $\hat{d}_{PMF}$ .

## 5. Conclusions and Future Work

Statistical analysis of time series has experienced significant growth during the last 50 years. Although the majority of works focus on real-valued time series, ordinal time series have received a great deal of attention during the present century. The R package **otsfeatures** is fundamentally an attempt to provide different functions allowing the calculation of well-known statistical quantities for ordinal series. As well as providing an useful description about the behavior of the time series, the corresponding quantities can be used as input for traditional data mining procedures, as clustering, classification, and outlier detection algorithms. Additionally, **otsfeatures** includes some tools enabling the execution of classical inferential tasks, as hypothesis tests. It is worth highlighting that several functions of the package can also be used to analyze ordinal data without a temporal nature. 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 ordinal datasets. In fact, the few software tools designed to deal with this class of databases focus on specific tasks (e.g., clustering or classification), application areas, or types of ordinal models. Package **otsfeatures** also incorporates two databases of financial time series and three synthetic datasets, which can be used for illustrative purposes. Although **otsfeatures** is rather simple, it provides the much-needed tools for the standard analyses which are usually performed before more complex tasks as modeling, inference, or forecasting.

A description of the functions available in **otsfeatures** is given in the first part of this manuscript 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 paper. In the second part of the manuscript, the use of the package is illustrated by considering several examples involving synthetic and real data. This can be seen as a simple overview whose goal is to make the process of using **otsfeatures** as easy as possible for first-time users.

There are three main ways through which this work can be extended. First, as **otsfeatures** 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, note that the statistical quantities available in **otsfeatures** are defined for univariate time series. However, multivariate time series [29,30] are becoming more and more common due to the advances in the storage capabilities of everyday devices. Thus, a software package allowing the computation of statistical features for multivariate ordinal series could be constructed in the future. Note that **otsfeatures** assumes that all the values of a given time series are known. Although this assumption is entirely reasonable, it can become too restrictive in practice, since some OTS can include missing values. In this regard, it would be interesting to implement some functions

aimed at properly correcting these values so that the computation of ordinal features is not negatively affected.

**Author Contributions:** Conceptualization, Á.L.-O. and J.A.V.; Methodology, Á.L.-O.; Software, Á.L.-O.; Project administration, J.A.V. All authors have read and agreed to the published version of the manuscript.

**Funding:** 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 Centre for Information and Communications Technology Research (CITIC). CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the support of research centers of the Sistema Universitario de Galicia (CIGUS). We would also like to acknowledge the support by the European Regional Development Fund (ERDF).

**Data Availability Statement:** The datasets used in this manuscript are provided in the package that the paper describes.

**Acknowledgments:** The authors thank the referees for their useful comments on an earlier draft of this article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Weiß, C.H.; Pollett, P.K. Binomial autoregressive processes with density-dependent thinning. *J. Time Ser. Anal.* **2014**, *35*, 115–132. [CrossRef]
2. Stoffer, D.S.; Tyler, D.E.; Wendt, D.A. The spectral envelope and its applications. *Stat. Sci.* **2000**, *15*, 224–253. [CrossRef]
3. Weiß, C.H. *An Introduction to Discrete-Valued Time Series*; John Wiley & Sons: Hoboken, NJ, USA, 2018.
4. Weiß, C.H. Distance-based analysis of ordinal data and ordinal time series. *J. Am. Stat. Assoc.* **2020**, *115*, 1189–1200. [CrossRef]
5. Weiß, C.H. Regime-switching discrete ARMA models for categorical time series. *Entropy* **2020**, *22*, 458. [CrossRef]
6. Kupfer, J.; Brosig, B.; Brähler, E. A multivariate time-series approach to marital interaction. *GMS Psycho-Soc. Med.* **2005**, *2*, Doc08.
7. Stadnitski, T. Time series analyses with psychometric data. *PLoS ONE* **2020**, *15*, e0231785. [CrossRef] [PubMed]
8. Paminger, C.; Frühwirth-Schnatter, S. Model-based clustering of categorical time series. *Bayes. Anal.* **2010**, *5*, 345–368.
9. Chen, C.W.; Chiu, L. Ordinal time series forecasting of the air quality index. *Entropy* **2021**, *23*, 1167. [CrossRef]
10. Bandt, C. Ordinal time series analysis. *Ecol. Model.* **2005**, *182*, 229–238. [CrossRef]
11. Weiß, C.H. Measuring Dispersion and Serial Dependence in Ordinal Time Series Based on the Cumulative Paired  $\phi$ -Entropy. *Entropy* **2021**, *24*, 42. [CrossRef]
12. López-Oriona, Á.; Weiß, C.; Vilar, J.A. Fuzzy clustering of ordinal time series based on two novel distances with financial applications. *arXiv* **2023**, arXiv:2304.12249.
13. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2021.
14. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
15. Christensen, R.H.B. Ordinal—Regression Models for Ordinal Data. R Package Version 2022.11-16. 2022. Available online: <https://CRAN.R-project.org/package=ordinal> (accessed on 15 May 2023).
16. Amatya, A.; Demirtas, H.; Gao, R. *MultiOrd: Generation of Multivariate Ordinal Variates*; R Package Version 2.4.3; R Foundation for Statistical Computing: Vienna, Austria, 2021.
17. Heredia-Gómez, M.C.; García, S.; Gutiérrez, P.A.; Herrera, F. Ocapis: R package for ordinal classification and preprocessing in scala. *Prog. Artif. Intell.* **2019**, *8*, 287–292. [CrossRef]
18. Rao, C.R. Diversity and dissimilarity coefficients: A unified approach. *Theor. Popul. Biol.* **1982**, *21*, 24–43. [CrossRef]
19. Weiß, C.H. A new class of autoregressive models for time series of binomial counts. *Commun. Stat.-Theory Methods* **2009**, *38*, 447–460. [CrossRef]
20. Ristić, M.M.; Weiß, C.H.; Janjić, A.D. A binomial integer-valued ARCH model. *Int. J. Biostat.* **2016**, *12*, 20150051. [CrossRef] [PubMed]
21. Kuhn, M. *Caret: Classification and Regression Training*; R Package Version 6.0-93; R Foundation for Statistical Computing: Vienna, Austria, 2022.
22. Lopez-Oriona, A.; Vilar, J.A. *mlmts: Machine Learning Algorithms for Multivariate Time Series*; R Package Version 1.1.1; R Foundation for Statistical Computing: Vienna, Austria, 2023.
23. Maechler, M.; Rousseeuw, P.; Struyf, A.; Hubert, M.; Hornik, K. *Cluster: Cluster Analysis Basics and Extensions*; R Package Version 2.1.2—For New Features, See the ‘Changelog’ File (in the Package Source); R Foundation for Statistical Computing: Vienna, Austria, 2021.

24. Campello, R.J. A fuzzy extension of the Rand index and other related indexes for clustering and classification assessment. *Pattern Recognit. Lett.* **2007**, *28*, 833–841. [[CrossRef](#)]
25. Mouselimis, L. *ClusterR: Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans, K-Medoids and Affinity Propagation Clustering*; R Package Version 1.2.6; R Foundation for Statistical Computing: Vienna, Austria, 2023.
26. Shimizu, Y. Multiple Desirable Methods in Outlier Detection of Univariate Data with R Source Codes. *Front. Psychol.* **2022**, *12*, 6618. [[CrossRef](#)]
27. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv. CSUR* **2021**, *54*, 1–33. [[CrossRef](#)]
28. Weng, X.; Shen, J. Detecting outlier samples in multivariate time series dataset. *Knowl.-Based Syst.* **2008**, *21*, 807–812. [[CrossRef](#)]
29. López-Oriona, Á.; Vilar, J.A. Outlier detection for multivariate time series: A functional data approach. *Knowl.-Based Syst.* **2021**, *233*, 107527. [[CrossRef](#)]
30. López-Oriona, Á.; Vilar, J.A. Quantile cross-spectral density: A novel and effective tool for clustering multivariate time series. *Expert Syst. Appl.* **2021**, *185*, 115677. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.