MASTER

MATHEMATICAL FINANCE

MASTER´S FINAL WORK

DISSERTATION

COMPARING THE CLUSTER EFFICIENCY OF FRAGMENTED-PERIODOGRAM AND FRAGMENTED-ACF

ANDREIA FILIPA MARTINS ALBINO

OCTOBER 2021

Lisbon School of Economics & Management
Universidade de Lisboa

MASTER
MATHEMATICAL FINANCE

MASTER´S FINAL WORK
DISSERTATION

COMPARING THE CLUSTER EFFICIENCY OF FRAGMENTED-PERIODOGRAM AND FRAGMENTED-ACF

ANDREIA FILIPA MARTINS ALBINO

SUPERVISION:
PROFESSOR DOUTOR NUNO PAULO DE SOUSA ARROBAS CRATO

OCTOBER 2021

Glossary

ACF - Autocorrelation Function

AR - Autoregressive model

ARIMA - Autoregressive Integrated Moving Average model

DFT - Discrete Fourier Transform

JEL - Journal of Economic Literature

Resumo

Big data faz, cada vez mais, parte do nosso dia a dia e tem um impacto cada vez maior na nossa sociedade. Tem entusiasmado os investigadores para encontrarem novos métodos para lidar com conjuntos de dados muito grandes. Na análise das séries temporais, a mesma necessidade surge. Há algumas décadas, os investigadores batalhavam para encontrar séries de dados longas o suficiente para as analisar; hoje em dia, estão frequentemente sobrecarregados com conjuntos de dados que são demasiado grandes para lidar com métodos tradicionais.

O objetivo deste trabalho é estudar e propor métodos para medir as semelhanças e diferenças entre diferentes séries temporais. Esses métodos devem ser computacionalmente simples e perdem, naturalmente, algumas informações, mas devem ser capazes de fornecer métricas que são adequadas para agrupar grandes conjuntos para extensas séries temporais.

Para ser mais específica, estudo um método muito recente chamado periodograma fragmentado e confirmo por simulação, a sua utilidade para fornecer uma medida simples que leva ao agrupamento correto das séries temporais em consideração. Esta replicação dá algum conforto quanto à correção do exercício e leva a uma descoberta interessante sobre as regras de fragmentação. Portanto, proponho um método equivalente no domínio do tempo, o ACF fragmentado, e confirmo por simulação a sua utilidade. Finalmente, comparo os dois métodos e concluo que o ACF fragmentado tem um desempenho comparável, senão superior.

As simulações são, naturalmente, limitadas a classes particulares de modelos, o que abre espaço para pesquisas futuras, como discuto na secção final.

Palavras chave: Big Data; Séries Temporais; Agrupamento de séries temporais; periodograma fragmentado; ACF fragmentado

JEL CODES: C32; C38; C63

Abstract

Big data is increasingly having a large impact on our society and prompting researchers to find novel methods to deal with very large data sets. In time series analysis, the same need arises. A few decades ago, researchers were struggling to find long enough data series for their analysis; nowadays, they are often overwhelmed with data sets that are too large to handle with traditional methods.

The aim of this work is to study and propose methods for measuring the similarities and dissimilarities among different time series. These methods should be computationally simple and naturally lose some information, but they should be able to provide metrics that are suitable for clustering large sets on long time series.

To be more specific, I study a very recent method called fragmented periodogram and confirm by simulation its usefulness to provide a simple measure that leads to correct clustering of the time series under consideration. This replication gives some comfort as to the correctness of the exercise and leads to an interesting finding regarding the fragmenting rules. Then, I propose an equivalent method in time domain, the fragmented ACF and confirm by simulation its usefulness. Finally, I compare the two methods and conclude that the fragmented ACF has a comparable if not superior performance.

The simulations are naturally limited to particular classes of models, which opens room for further research, as I discuss in the final section.

Keywords: Big Data; Time Series; Time Series Clustering; Fragmented-periodogram; Fragmented-ACF

JEL CODES: C32; C38; C63

Contents

List of Figures

List of Tables

Agradecimentos

Ao Professor Nuno Crato, por todo o apoio e disponibilidade que demonstrou ao longo destes meses, em especial dadas as barreiras vividas devido às contingências atuais. O seu entusiasmo e a sua empatia contagiantes, tornaram este meu projeto final, numa jornada otimista, mas não menos desafiante.

Um obrigada aos meus amigos e namorado, por todo o carinho e pela forma como contribuem, diariamente, para a minha evolução, fazendo-me sentir realizada todos os dias. Aproveito para agradecer, em especial, à Ana Sofia Moutoso, ao Tomás Carneiro e ao Bernardo Covas, pela partilha dos seus conhecimentos, que desempenharam um papel fulcral na resolução de obstáculos com os quais me deparei.

O meu maior obrigado à minha família, primordialmente aos meus pais, por todo o apoio incondicional ao longo de 25 anos, e por me darem a possibilidade de seguir o caminho que acredito ser o melhor para o meu futuro. Graças a vocês, poderei ter uma vida e um futuro risonho!

Obrigada!

1 Introduction

Nowadays, there is an increasing need to automate statistical estimation methods minimizing the loss of information. Furthermore, it is essential to make these methods as automatic as possible, fast and efficient.

Data analysis has gained increasing weight in society and is still a topic of great focus when it comes to business and scientific levels. It is crucial that all data are as well analyzed as possible, which has raised many challenges for researchers. Fortunately, this is an increasingly talked about and improved topic. Therefore, in addition to the fact that data analysis is well performed, it is also necessary that it be performed in the shortest possible time.

In this work, I will deal with a problem of big data analysis by studying a particular problem which has recently received much attention. I will compare two methods of grouping or clustering time series. The first will be a particular spectral method, the so-called fragmented-periodogram clustering suggested in Caiado, Crato, and Poncela (2020). This method does not imply the computation of the full periodograms but only of the periodogram components around the frequencies of interest. It then compares the periodogram ordinates for the various time series and groups them with common clustering methods.

The second method will be using the Autocorrelation Function (ACF). As with the spectral method, this method does not perform the computation of the full ACF, but only of the ACF components around the lags of interest. It then compares the selected ACF values for the various time series and groups them by using common clustering methods. Thus, I call it a fragmented-ACF approach.

A survey on time series clustering can be seen in Caiado, Crato, and Poncela (2020) and references therein, and it was somehow my inspiration for this work. The first method has been suggested and studied. However, as far as I know, the corresponding idea for the ACF has not been yet suggested and studied.

In this context, I will compare the first method, the fragmented-periodogram, with the second method, the fragmented-ACF. Then, I will compare the efficiency of grouping data with the fragmented-periodogram and with the fragmented-ACF.

The plan for the rest of the work is as follows. Section 2 reviews the relevant literature including the definition of the periodogram, and the ACF, and its evolution, as well as the research regarding fragmented and big data approaches. Section 3 seeks to present and fully specify the methodology and data used in this work. It includes a descriptive analysis, in which some technical features and statistics of the series will be analyzed. Section 4 compares the models within each method and, after choosing the best options, make a comparison between them and report the corresponding results. It will be presented an explanation of the Python script used. Section 5 concludes discussing the limitations of the methods used and referring to possible further investigations in the field.

2 Literature Review

2.1 The problem of clustering big data time series

Enterprises and public organizations collect very large data sets, which can be structured, semistructured and unstructured. This immense combination of data is known as big data. Such data is often used in several advanced analytic applications, forecasts and is also used in machine learning projects.

Laney et al. (2001) identified three main characteristics of big data, known for three V's as described below:

1. the *volume* regarding the size of data;
2. the *variety* as different types of data from several sources;
3. the *velocity* at which the data collected is processed.

After that, a lot of other V's were added to these three standard V's such as, *value*, *variability* and *veracity*.

Companies use big data to improve operations, provide better customer service, create personalized marketing campaigns and take other actions that, ultimately, can increase revenue and profits. As a result, businesses that use it effectively hold a potential competitive advantage over those that don't because they can take faster and more informed business decisions.

There are several examples where organizations use big data. First example, in the energy industry, in which big data helps identify potential drilling locations and monitor pipeline operations; likewise, utilities use it to track electrical grids. Second, financial services firms use big data systems for risk management and real-time analysis of market data. Third, manufacturers and transportation companies rely on big data to manage their supply chains and optimize delivery routes and, many other examples can be given.

The term "big data" is relatively recent. It will undoubtedly be an intense journey in the scientific for understanding how to treat extensive data, and their characteristics so that they can be treated.

Big data problems require making several tradeoffs among desired scalability, availability, performance, and security. For some problems, precise solutions are intractable and may require faster and approximated algorithms that run the risk of decreasing the quality of the solution.

Kirlić et al. (2017) showed that there are several ways to deal with big data and there is even a wide range of literature with various methods. I will cover in more detail and technique a procedure to compare and group big data time series.

There are considerable time series research efforts to find a solution to various types of problems in this type of data. One of the several problems is finding and extracting similarities from the extensive big data repositories, in order to overcome the well-known challenges of big data management. For this problem, Agrawal et al. (1993) proposed an indexing method for time sequences for processing similarity queries by using the Discrete Fourier Transform (DFT), taking into account the most substantial first frequencies of the frequency domain. Furthermore, they highlight Parseval's theorem, which describes the Fourier transform preserving the Euclidean distance in the time or frequency domain. After having the map sequence, they used R^*-trees to index the sequences and efficiently answer similarity queries.

Another interesting study still within the topic of finding similarities or patterns in time series with spectral methods can be found on Chan et al. (2003), *Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping*, where they focused on two metrics, namely, Euclidean distance and time warping distance. A traditional method uses the means of DFT, reducing the dimension of the sample. The authors used a technique called Haar Wavelet Transform and proposed the use of proper normalization in order to guarantee no cut for Euclidean distance. They found that this technique has competitive performance for their experiments. Furthermore, time warping distance can handle the time shifts of patterns, unlike Euclidean distance. Since this time warping distance is not metric and is very complex to compute, they suggested a Haar wavelet-based approximation function for time warping distance, called Low Resolution Time Warping, which trades off a small amount of accuracy in fewer computations. This research helped define a similarity measurement during the matching process to depict the similarity between two time series.

A lot of other studies can be seen around this topic of identifying similarities in time series (see, Bettini et al. (1998), Berndt et al. (1996))

Moving forward, another fundamental issue in time series is subsequence searching in time series. In this topic Fancoua et al. (1996) introduced a new technique for the competitive identification of piecewise stationary time series were quoting, " A neighborhood map of one step predictors competes for the data during training. The winner is granted the largest parameter update, while other predictors are allowed smaller updates, decreasing with distance from the winner on the neighborhood map. In addition to performing piecewise segmentation and identification, the technique maps similar segments of the time series as neighbors on the neighborhood map." The simplicity of this approach is that the predictors operate in parallel and can be trained in their usual way.

When comparing more than two time series, one may want to group them according to the similarity or dissimilarity revealed with the chosen metric. Various methods of clustering have been extensively discussed in the literature.

Cuzzocrea (2020) provides a recent review of clustering methods. One of the most used is the *k-mean clustering* reported by Jancey (1966). The global k-means algorithm is described as follow:

Suppose a data set $X = x_1, ..., x_N, x_N \in R^d$. The M-clustering problem aims at partitioning this data set into M disjoint subsets (clusters) $C_1, ..., C_M$, such that a clustering criterion is optimized. The most widely used clustering criterion is the sum of the Euclidean distances between each data point x_i and centroid m_k (cluster center) of the the subset C_k which contains x_i. This criterion is called clustering error and depends on cluster centers $m_1, ..., m_M$:

$$E(m_1, ..., m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) \|x_i - m_k\|^2,$$

where $I(X) = 1$ if $x_i \in C_k$ is true and 0 otherwise.

The k-means algorithm finds locally optimal solutions for the clustering error. It is a fast iterative algorithm that has been used in many clustering applications. It is a point-based clustering method that starts with the cluster centers initially placed at arbitrary positions and proceeds by moving at each step of the cluster centers to minimize the clustering error. The main disadvantage of the method lies in its sensitivity to the initial positions of the cluster centers. Therefore, in order to obtain near optimal solutions using the k-means algorithm, several runs must be scheduled differing in the initial positions of the cluster centers.

Caiado, Crato, and Poncela (2020) proposed a new method for comparing and clustering long and varied time series, the so-called fragmented-periodogram approach. These authors develop and study a new frequency-domain procedure for characterizing and comparing large sets of extensive time series. Instead of using all the information of the data, which would be computationally very expensive, they propose some regularization rules to select and summarize the most relevant information for clustering purposes. The authors proposed to use a fragmented periodogram computed around the driving cyclical components of interest and to compare the various estimates. This procedure is computationally simple and straightforward to summarize relevant information of the time series. These authors also provide a simulation exercise which shows that a smoothed fragmented periodogram works better than the nonsmoothed one and not worse than the complete periodogram for medium to large sample sizes.

2.2 Metric definition to data cluster

To do any clustering in time series, I must define measures of similarity of dissimilarity. It is essential to understand that not every distance measure is a metric. To certify as a metric, a function d must satisfy the following conditions:

Let x and y be any two objects in a set and $d(x,y)$ be the distance between x and y.

1. The distance between any two points must be non-negative, that is, $d(x,y) \geq 0$.

2. The distance between two objects must be zero if and only if the two objects are identical, that is, $d(x,y) = 0$ is and only if $x = y$.

3. Distance must be symmetric, that is, distance from x and y is the same as the distance from y to x, that is, $d(x,y) = d(y,x)$

4. The measure must satisfy the triangle inequality, which is $d(x,z) \leq d(x,y) + d(y,z)$.

A metric often used is the Euclidean distance. Euclidean distance is a standard metric for geometrical problems. It is the ordinary distance between two points and can be easily measured with a ruler in two- or three-dimensional space. Euclidean distance is widely used in clustering problems[1]. It satisfies all the above four conditions and therefore is a valid metric. It is also the default distance measure used with the K-means algorithm. In $\mathbb{R}$ the Euclidean distance between two vectors $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ is always defined. It can be calculated as:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Most vector spaces in machine learning belong to this category. When we conduct machine learning tasks, we can usually measure Euclidean distances in a dataset during preliminary data analysis.

Another metric often used is cosine similarity. Cosine similarity between two vectors is the cosine of the angle they form in an inner product space and corresponds to their dot product divided by the product of their magnitudes. For example, if x and y are vectors, the cosine similarity is:

$$\cos(\theta) = \frac{x \cdot y}{||x|| \, ||y||}$$

The first of this type of metrics was introduced in time series by Piccolo (1990). His proposal is simply the Euclidean distance between the coefficients of the autoregressive (AR) representation of the time series under consideration. Taking into account two time series allowing AR representations, x_t and y_t, with t integer, i.e.,

$$x_t = \phi_{1,x} x_{t-1} + \phi_{2,x} x_{t-2} + \phi_{3,x} x_{t-3} + ... + \varepsilon_t$$

[1]Chauhan et al. (2013)

$$y_t = \phi_{1,y} y_{t-1} + \phi_{2,y} y_{t-2} + \phi_{3,y} y_{t-3} + \ldots + \varepsilon_t$$

Therefore, the distance showed by Piccolo (1990) is as follow,

$$d(x,y) = \sqrt{\sum_{j=1}^{\infty} (\phi_{j,x} - \phi_{j,y})^2} \tag{1}$$

If the autoregressive coefficients are square summable, then this distance exists. Nonetheless, even if the series are not stationary, it is possible to use truncated AR representations, which are always possible for empirical time series. Piccolo's method requires the estimation of a model and the computation of its AR coefficients. The definition of a distance between time series models immediately allows applications to clustering algorithms.

Later, Diggle et al. (1991) proposed a non-parametric approach to compare the spectrum of two time series based on the underlying cumulative periodograms. They suggested *normalized* cumulative periodograms, as they wanted to detect only shape differences between the two underlying spectra. Let $I_x(\omega)$ and $I_y(\omega)$ denote the periodogram ordinates of $x_t : t = 1, \ldots, n$ and $y_t : t = 1, \ldots, n$ respectively, each evaluated at frequencies ω of the form $\omega_j = 2\pi j/n$, where $j = 1, \ldots, m$ and $m = [(n-1)/2]$, i.e,

$$I_x(\omega) = (2\pi n)^{-1} \left| \sum_{t=1}^{n} x_t e^{-i\omega t} \right|^2$$

analogous expression for $I_y(\omega)$. For x, the normalized cumulative periodogram is

$$F_x(\omega_j) = \sum_{i=1}^{j} I_x(\omega_i) / \sum_{i=1}^{m} I_x(\omega_i)$$

and similarly for $F_y(\omega_j)$.

As already discussed, in 2020, Caiado, Crato, and Poncela (2020) proposed *fragmented-periodogram*. For the *fragmented-periodogram* definition, let us consider an s as the frequencies of a periodogram, and let us keep the notation used by the authors. The proposal was computing the periodogram but only around the frequencies s of interest. As the authors explain, it is not possible to choose symmetric intervals both in the time domain and in the frequency-domain. Willing to use symmetry in the time domain, it is not possible to simply construct an interval of amplitude $2h$ around the desired frequency. This would make the time-domain interval asymmetrical. The issue, can be perfectly described by figure 1.

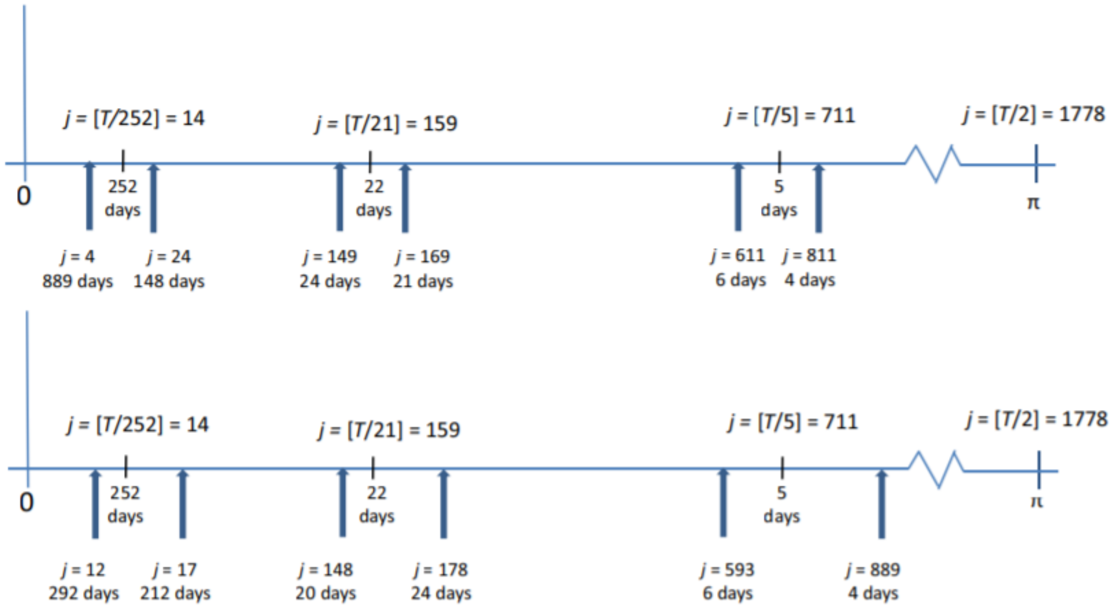


Figure 1: Illustration of limits for fragmented periodogram frequencies with two windows: frequency-domain symmetric (top) and time-domain symmetric (bottom). It is clear that symmetry in one domain implies asymmetry in the other domain, where T is the sample size of the series in study (Caiado, Crato, and Poncela, 2020)

In their paper, they proposed a lower and an upper bounds, l and u, respectively, as

$$l = \left\lfloor \frac{T}{s+h_s} \right\rfloor \tag{2}$$

and

$$u = \left\lceil \frac{T}{s-h_s} \right\rceil \tag{3}$$

where $\lfloor\ \ \rfloor$ represents the floor function, $\lceil\ \ \rceil$ the ceiling function, T, the series sample size, s, the seasonal periods desired to be analyzed, and $h_s = [\frac{s}{4}]$. Considering these bounds, it will be possible to create acceptable ranges to fragment the periodogram and in this way to get the main goal of the paper which was creating a method that works in the frequency domain, uses a computationally simple and very parsimonious approach.

Within the topic of methods for clustering time series, a different metric was proposed by Galeano et al. (2001). This metric is based on the estimated autocorrelation function (ACF). Admitting that I have a set of time series $X = (x_{1,t}, ..., x_{k,t})'$ and $\hat{\rho} = (\hat{\rho}_{i,1}, ..., \hat{\rho}_{i,m})$ is the vector of the estimated autocorrelation coefficients of the time series i for some m, s.t, $\hat{\rho}_j \cong 0$ for $j > m$. The distance between x and y time series is described by

$$d_{ACF}(x,y) = \sqrt{(\hat{\rho}_x - \hat{\rho}_y)'\Omega(\hat{\rho}_x - \hat{\rho}_y)}, \tag{4}$$

where Ω is a diagonal matrix of positive elements. When $\Omega = I$ (identity matrix) the distance between the ACF coefficients of the time series x and y is equal to the Euclidean distance. On the other, when $\Omega = [cov(\hat{\rho})]^{-1}$ is the inverse covariance matrix of the autocorrelations, obtains the Mahalanobis distance between the autocorrelations. To use weights that decrease with the autocorrelation lag is very common as well.

After that, Caiado, Crato, and Peña (2006) suggested to use the frequency domain. They demonstrated their approach in different contexts. First, they found the similarities and dissimilarities in every pair of time series using 7 different metrics. The second step was grouping time series into two different clusters, depending on stationary or nonstationary time series, using good agglomerative hierarchical clustering algorithm as the single linkage, complete linkage, or the average linkage, i.e, maximizes the minimum distance between objects in the same group, minimizes the maximum distance between objects in the same group and averages the distance between objects in different groups, respectively). They thought of a different approach also using a non-hierarchical clustering procedure, such as the k-means algorithm. This implementation of the k-means clustering algorithm was based on Euclidean distances between standardized observations, autoregressive weights, autocorrelation coefficients, partial autocorrelation coefficient, inverse autocorrelation coefficients, and normalized periodogram ordinates in the logarithm scale.

The spectrum and ACF coefficients are often defined for stationary processes, although the definitions can be extended for integrated processes (Peña et al., 2006).

Following Caiado, Crato, and Peña (2006) and Caiado, Crato, and Poncela (2020), spectral methods handling the periodogram described for each frequency $w_j = \frac{2\pi j}{T}$, $j = 1,...,[T/2]$

$$I_x(w_j) = T^{-1}\left|\sum_{t=1}^{T} x_t e^{itwj}\right|^2 \tag{5}$$

and figure out distance among time series x and y, described as follow

$$d(x,y) = \sqrt{\sum_{j=1}^{[T/2]} (P_x(\omega_j) - P_y(\omega_j))^2}, \tag{6}$$

where P may represent the periodogram, I, the normalized periodogram $\gamma_0^{-1}I$ where γ_0 is the variance of the series, or the log-normalized periodogram $ln(\gamma_0^{-1}I)$.

Caiado, Crato, and Peña (2006) showed that using the normalized periodogram works very well to discriminate nonstationary and near-stationary time series.

Later, Caiado, Crato, and Poncela (2020) proposed to work with specific characteristics of the series dynamics that define the fundamental fluctuations. This way, they intended significantly to cut down the computations when there is a considerable number of observations. Instead of computing and working with the whole periodogram, they propose to fragment that periodogram, computing the periodogram only around the frequencies s of interest.

After suggesting the fragmented the periodogram, the authors also questioned themselves of "how can periodogram smoothing contribute to better results?". And the question was positively answered by smoothing the fragmented periodogram, which can really improve the results. They showed that for both the periodogram and the log-periodogram, smoothing reduces the variance of the differences. This established an argument in favour of smoothing the fragmented periodogram. The whole suggested procedure is as follows.

1. Compute the periodogram, normalized periodogram or log normalized periodogram only for the ordinates in the intervals

$$\left[\frac{2\pi}{s+h_s}; \frac{2\pi}{s-h_s}\right]$$

where h_s depends on the frequencies s of interest.

2. Smooth the fragmented periodogram, normalized periodogram, or log normalized periodogram P^{frag}. They suggested the two most popular smoothers, the Bartlett and the rectangular, and noted that any other smoother could be used.

   Bartlett smoother:
$$\hat{P}_j^k = \frac{1}{M}\sum_{i=-M}^{M}\left(1-\frac{|i|}{M}\right)P_{j-i}^{frag} \tag{7}$$

   Rectangular smoother:
$$\hat{P}_j^k = \frac{1}{2M+1}\sum_{i=-M}^{M}P_{j-i}^{frag}$$

   where $k = 2M+1$

2.3 New proposal

The idea of only computing and using part of the periodogram can be easily translated in the time domain analysis. We can also compute part or parts of the ACF sample function and use the pointwise comparison to define a measure of similarity.

To be specific, consider as before that we are interested in particular points or areas of the autocorrelation. We may, for instance, suspect that the series under comparison are driven by seasonal factors of similar lags, e.g., weekly or monthly lags.

Then, we can compute the sample ACF functions of the series under consideration only around the lags of interest and arrive at the following[2] measure of similarity:

Considering a stationary process Z_t, its mean $E(Z_t) = \mu$ and the variance $Var(Z_t) = E(Z_t-\mu)^2 = \sigma^2$, which are constant and the covariance $Cov(Z_t, Z_s)$, which are functions only of the time difference $|t-s|$. For this specific case, the covariance between Z_t and Z_{t+k} is as follow,

$$\gamma_k = Cov(Z_t, Z_{t+k}) = E(Z_t-\mu)(Z_{t+k}-\mu)$$

[2]Wei (2006)

and the correlation between Z_t and Z_{t+k} as

$$\rho_k = \frac{Cov(Z_t, Z_{t+k})}{\sqrt{Var(Z_t)\sqrt{Var(Z_{t+k})}}} = \frac{\gamma_k}{\gamma_0} \tag{8}$$

Then, for a stationary process, the autocovariance function γ_k and the autocorrelation function ρ_k have the following properties:

1. $\gamma_0 = Var(Z_t); \rho_0 = 1$
2. $|\gamma_k| \leq \gamma_0; |\rho_k| \leq 1$
3. $\gamma_k = \gamma_{-k}$ and $\rho_k = \rho_{-k}$ $\forall k$, i.e., γ_k and ρ_k are even functions and for those reasons, symmetric about the lag $k = 0$. Basically, the time difference between Z_t and Z_{t+k} and Z_t and Z_{t-k} are the same.
4. The autocovariance function γ_k and the autocorrelation function ρ_k are positive semidefinite
$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \gamma_{|t_i - t_j|} \geq 0$$
and
$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \rho |t_i - t_j| \geq 0$$
$\forall t_1, t_2, ..., t_n$ and $\forall \alpha_1, \alpha_2, ..., \alpha_n$.

After computing the ACF, to find and extract the similarities of two times series, x and y, only around s of interest, we figure out the distance between parts of autocorrelation in study, as follow

$$d_{fragmented-ACF}(x,y) = \sqrt{\sum_{i \in s} (\rho_{i,x} - \rho_{i,y})^2}$$

3 Assessment of the fragmentation methods by simulation

After reading the existing literature review and directing the topics to our work of interest, I follow as main references Piccolo (1990), Caiado, Crato, and Peña (2006), and Caiado, Crato, and Poncela (2020).

3.1 First Simulation Exercise: Testing for periodogram

In this subsection I will study the first clustering method approach of this work by doing simply what was proposed in Caiado, Crato, and Poncela (2020).

In this context, I have performed several simulations. For $N = 4000$, where N is the number of replications of time series, I have simulated the following model

$$y_t = \phi_w y_{t-w} + \phi_m y_{t-m} + \phi_a y_{t-a} + \varepsilon_t$$

where ϕ_w is the AR parameter associated to the weekly cycle, ϕ_m is the AR parameter associated to the monthly cycle, ϕ_a the AR parameter associated to the annual cycle, and ε_t is white noise. In each run, the $N = 4000$ time series were divided into 2 groups considering two different data generating processes. For the first group of series the lags associated to the weekly, monthly, and annual seasonal cycles were 5, 21, and 252, respectively, while for the second group, the lags associated with the weekly, monthly, and annual seasonal cycles were 4, 25 and 300. Then, I obtained the first set of simulations data as follow

$$y_t = \phi_w y_{t-5} + \phi_m y_{t-21} + \phi_a y_{t-252} + \varepsilon_t \tag{9}$$

which will be compared with the second set of data generated by

$$y_t = \phi_w y_{t-4} + \phi_m y_{t-25} + \phi_a y_{t-300} + \varepsilon_t \tag{10}$$

After this, it was computed the periodogram by following the equation (5) and then, log normalized the periodogram ($ln(\gamma_0^{-1} I)$). All the work was performed in a log normalized periodogram, and, for that reason, from now on, the log normalized periodogram will be called just periodogram.

There will be two other versions: the fragmented and the smoothed fragmented periodogram. For the smoothed version, I decided to use the Bartlett filter since it worked very well in Caiado, Crato, and Poncela (2020), and this filter can be computed by equation (7) with $M = \frac{u-l}{2}$, u and l as defined in equation (2) and (3), respectively. For h_s I considered 0.5, 3 and 45 for annual, monthly and weekly cycles, respectively. Using these values for h_s I guarantee that the same percentage of ordinates are used in fragmented versions of the periodogram. Finally, I have clustered the time series, computing the Euclidean distance of each pair, choosing the minimum distance from all the combinations of each 2 groups of each data process (9) and (10), of the three approaches (for all periodogram, fragmented-periodogram and smoothed fragmented-periodogram). I have considered sample sizes of $T = 500, 1000, 2000, 5000$ and 10000 with 1000 replications of each data generated process.

Tables of this clustering method will be shown below. These tables can be read in the same way. The first column (T, p, f) shows as T the sample size, p the number of ordinates of the periodogram, and f as the number of ordinates that I use in the fragmented and smoothed fragmented periodograms, which are the same. Then the table shows other columns, each one of them, where I show the percentage of times that the N time series are correctly clustered for each sample size T when using the entire periodogram, fragmented and smoothed fragmented versions. By correctly clustered, I mean that the minimum Euclidean distance described above comes from the two series of the same model.

I want to check if using only a small fraction of the information (ordinates in the periodogram), I could get classification results comparable to those obtained when using the whole periodogram. I would also like to see the effect of smoothing the fragmented periodogram before clustering. The overall effect of smoothing the periodogram is to reduce the variance of the spectral estimates differences as the number of ordinates used for smoothing increases. The effect of smoothing should be larger; the larger the interval I use for smoothing. In this sense, I would like to point out that the number of ordinates associated to the weekly cycle in the fragmented periodogram is larger than that related to the monthly cycle and both larger than that used in the annual cycle.

Table 1 shows the results of simulation when I assign the same value to the autoregressive parameters associated to each one of the seasonal cycles, so all the cycles rely on the exact value of the parameters and it is not the difference in the value of the parameters what drives the good or bad performance of the method. I decided to group by sets as authors of Caiado, Crato, and Poncela (2020) did. So, the first column shows the sample size and information regarding the number of ordinates used, as explained above. From column 2 to 4, 5 to 7 and 8 to 10, the percentage of correctly classified for each set of parameters used in the simulation. As expected, results improve as we move down, that is when I increase the sample size T. Moreover, for a given sample size, the best results are always given when using the whole periodogram. Although the smoothed periodogram, which uses a small percentage of information, follows closely the outcome of the full periodogram for moderate to large sample sizes. The fragmented periodogram gives the worst results of all the procedures but it works pretty well from $T = 2000$. As smoothing decreases the variance of the periodogram, it could detect differences in periodograms more efficiently using the smoothed version than just the fragmented version.

As expected, the results presented in columns 2 to 4 are better than 5 to 7 and even better than 8 to 10. Let's give a little attention to the numbers from the last set of simulations (columns from 8 to 10) where for each of the models, the coefficients are assigned to 0.1. The results are less satisfactory since the processes are close to white noise.

Moving on to table 2, for a given set of parameters, the results continually improve with the sample size T. Similarly, to table 1 there exists the same scenario; working with the whole periodogram, I got better results in classifying time series, followed by smoothed periodogram and finally by the fragmented-periodogram.

Another interesting simulation is to see how could be the results if I directly choose the lags to consider in fragmented-periodogram, and so not considered the boundaries (2) and (3). In this way, the number of ordinates will be the same, whatever the sample size T. Table 3 and Appendix A.1 are the results of this new simulation to test for specific boundaries around the s. Notice that, since the simulation is around of s of interest does not make sense

Table 1: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for the periodogram when $\phi_w = \phi_m = \phi_a = .3$, $\phi_w = \phi_m = \phi_a = .2$ and $\phi_w = \phi_m = \phi_a = .1$

(T, f, p)	$\phi_w = .3$, $\phi_m = .3$, $\phi_a = .3$			$\phi_w = .2$, $\phi_m = .2$, $\phi_a = .2$			$\phi_w = .1$, $\phi_m = .1$, $\phi_a = .1$		
	All	Frag	Smth.	All	Frag	Smth.	All	Frag	Smth.
$(500, 250, 29)$	95.9	56.3	79.7	72.9	43.2	63.5	45.1	36.1	44.9
$(1000, 500, 56)$	99.9	76.1	91.2	90.5	54.0	75.0	52.4	38.2	45.9
$(2000, 1000, 112)$	100.0	91.0	99.8	98.6	65.2	93.1	59.7	43.0	59.0
$(5000, 2500, 279)$	100.0	99.3	100.0	100.0	83.8	100.0	77.7	49.0	78.6
$(10000, 5000, 558)$	100.0	100.0	100.0	100.0	96.1	100.0	91.8	54.5	93.6

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 2: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for the periodogram when $\phi_w = .4, \phi_m = .3, \phi_a = -.2$, $\phi_w = .4, \phi_m = -.3, \phi_a = .2$ and $\phi_w = -.4, \phi_m = .3, \phi_a = .2$

(T, f, p)	$\phi_w = .4$, $\phi_m = .3$, $\phi_a = -.2$			$\phi_w = .4$, $\phi_m = -.3$, $\phi_a = .2$			$\phi_w = -.4$, $\phi_m = .3$, $\phi_a = .2$		
	All	Frag	Smth.	All	Frag	Smth.	All	Frag	Smth.
$(500, 250, 29)$	99.5	60.9	85.1	99.2	60.1	87.3	99.8	66.4	94.2
$(1000, 500, 56)$	100.0	74.1	94.0	100.0	68.5	96.2	100.0	82.3	97.8
$(2000, 1000, 112)$	100.0	86.3	99.6	100.0	78.9	99.8	100.0	92.4	100.0
$(5000, 2500, 279)$	100.0	97.6	100.0	100.0	94.7	100.0	100.0	99.2	100.0
$(10000, 5000, 558)$	100.0	99.9	100.0	100.0	99.1	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

to compute the "All" periodogram, as studied before. I find it interest to see what will be the results when it joins the annual cycle of the model (9) and the annual cycle of the other model (10) in one interval. So, this can be seen from column 2 to 3, and the weekly and monthly cycles are also represented in this simulation. From column 4 to 5 are represented the weekly, monthly cycles, and the annual cycle; this last cycle is divided into two intervals, one around the $s = 252$ and the other around the $s = 300$ of interest. It will also be interesting to see the differences in the efficiency of the cluster among these two approaches. Finally, from 6 to 7 are represented the results when we consider only the weekly and monthly cycles to realize how the annual cycles can influence the efficiency of the cluster. From column 8 to 9, the simulation considers only the annual cycles of both models (model (9) and model (10)).

Analyzing these new simulations, table 3 and tables presented in appendix A.1, show that the smoothed fragmented periodogram has a better performance when comparing only with the fragmented-periodogram. The results are reasonable because the smoothed periodogram has less variance comparing with no-smoothed. In the end, in the first simulation when used the boundaries represented in 2 and 3, the results were much better than this new one.

Other simulations for other coefficients values were made and can be seen in Appendix A.1.

Table 3: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for the periodogram when $\phi_w = \phi_m = \phi_a = .3$

	$\phi_w = \phi_m = \phi_a = .3$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	40.6	60.5	40.6	60.5	44.4	47.7	36.3	60.3
1000	72.5	79.0	61.5	71.8	39.8	40.7	64.1	75.0
2000	72.9	95.8	62.4	88.1	50.4	60.8	59.2	85.0
5000	79.4	98.8	53.4	87.6	53.2	80.7	39.8	63.2
10000	74.1	98.8	65.8	90.6	38.5	51.8	69.8	93.8

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

3.2 Second Simulation Exercise: Testing for ACF

In the next set of simulations, I will test the ACF method to finally compare both methods, spectral method and ACF method, to see which of them are more efficient by cluster the data with Euclidean Distance as did for the periodogram.

To be able to compare both estimators (spectral and ACF), my Supervisor and I thought of a *fragmented-ACF* approach, which is precisely the Autocorrelation function, as defined in (8), but fragmented into the lags k of interest for the specific study.

Going forward on the simulations, the so-called method of ACF was run similarly to the previous method (the spectral); $N = 4000$ time series were divided into 2 groups considering two different data generating processes, one set by model (9) and another set by model (10). For a better comparison, the same data were generated throw this method and the spectral. After this, it was computed the Autocorrelation Function of each time series by following equation (8) and another version of the fragmented-ACF, which is to consider the lags s of interest. Finally, I have clustered the time series computing the *Euclidean distance* of each pair choosing the minimum distance from all the combinations of each 2 groups of each data process (9) and (10) of the two approaches (ACF and fragmented-ACF).

Notice that for the periodogram was used the frequency-domain and now, I propose an equivalent method in the time domain for the fragmented-ACF. For that reason, we thought of a heuristic method and decided to guarantee only that the number of ordinates used in the spectral method would also be used the same number in this method of ACF. Bartlett's filter will not be used in this method since it only makes sense to filter on Bartlett in the periodogram.

Following, I will present some results regarding this new method *fragmented-ACF*. Tables 4 and 5 next presented can be read in the same way. The first column shows (T, p), T the sample size, and p the number of values used in *fragmented-ACF*. Next to the sample size, there are other columns, each one of them, where I show the percentage of times that N time series are correctly clustered for each sample size T when using the number of lags of p. Looking at tables 4 and 5, there are common things between them. Interestingly, the efficiency of the cluster is better when considering only values around the s of interest, rather than considering compute the distance using the entire ACF. Nonetheless, similar to the spectral method, when the sample size increased, the results also improved.

Table 4: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for the ACF when $\phi_w = \phi_m = \phi_a = .3$, $\phi_w = \phi_m = \phi_a = .2$ and $\phi_w = \phi_m = \phi_a = .1$

(T, p)	$\phi_w = .3$ $\phi_m = .3$ $\phi_a = .3$		$\phi_w = .2$ $\phi_m = .2$ $\phi_a = .2$		$\phi_w = .1$ $\phi_m = .1$ $\phi_a = .1$	
	All	Frag	All	Frag	All	Frag
$(500, 29)$	93.7	99.8	84.9	97.0	55.0	70.9
$(1000, 56)$	96.3	100.0	95.2	99.9	63.7	91.0
$(2000, 112)$	97.9	100.0	99.2	100.0	81.3	98.4
$(5000, 279)$	98.2	100.0	100.0	100.0	95.5	100.0
$(10000, 558)$	99.3	100.0	100.0	100.0	99.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 5: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = .4, \phi_m = .3, \phi_a = -.2$, $\phi_w = .4, \phi_m = -.3, \phi_a = .2$ and $\phi_w = -.4, \phi_m = .3, \phi_a = .2$

(T, p)	$\phi_w = -.4$ $\phi_m = .3$ $\phi_a = .2$		$\phi_w = .4$ $\phi_m = -.3$ $\phi_a = .2$		$\phi_w = .4$ $\phi_m = .3$ $\phi_a = -.2$	
	All	Frag	All	Frag	All	Frag
$(500, 29)$	99.1	100.0	94.6	99.7	93.6	99.5
$(1000, 56)$	99.2	100.0	96.4	100.0	93.8	99.8
$(2000, 112)$	99.5	100.0	98.2	100.0	95.0	100.0
$(5000, 279)$	99.9	100.0	99.8	100.0	97.7	100.0
$(10000, 558)$	100.0	100.0	100.0	100.0	99.7	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Now, consider another simulation approach for fragmented-ACF, similarly what was made for the spectral method, where this approach is to fragment the ACF around the s of interest, narrowing the range of the fragments and maintain the number of values used throughout the simulation, even if I increase the sample size. The tables 6, 7, 8, 9, 10, 11 and

12 next presented can be read in the same way and are related to this new simulation. The first column shows T, the sample size. Next to the sample size, there are other columns, each one of them, where it showed the percentage of times that N time series are correctly clustered for each sample size T when using the lags showed in the second row. These position lags are the same used in the spectral method but are now adapted to ACF lags.

As seen in table 6, the percentage of times that N series are correctly clustered is relatively high. An overall, the event is cross-cutting in all columns; when the sample size increase, the event of correctly clustering the series is 100% effective. Nonetheless, when focusing on the sample size $T = 500$, I obtained the worst scenario on the distant lags (considering only the annual cycle, column 5) compared with the closer lags (considering the weekly and monthly cycles, column 4). When joining all the cycles, the weekly, monthly, and annual in a simulation, I got better results when I split the range of the annual cycle by a window around 252 lag and 300 lag (column 3) compared to the range where these annual cycles are merged (column 2). This can be explained by the specificity of the simulation, taking into account the lags of the models.

Looking at table 7, the results continue with a large percentage of successes. However, a slight decrease can be seen. Similar to table 6 the rate of efficiency of cluster series increase when the sample size increase. The worst scenario is noted in the last column when I consider only the annual cycle in the simulation. The best scenario remains where I include all cycles, but the annual cycle is split in two.

Analyzing table 8, this is the case where I want to check the results of the simulations when the coefficients of both models are small and, therefore, close to the case of white noise. As expected, the results are worst than in table 7 and table 6 due to the proximity to white noise. Nonetheless, the results are pretty surprising because I obtained results quite high, and I even got 100% of success in some cases and sample sizes.

For the next group of tables, I will test the case when each coefficient of the models is going to have different weights for each cycle.

In this scenario, the percentage of success for efficiently cluster data is excellent. We obtained 100% of data correctly clustered, approximately in all cases, when the data size starts from 1000. However, in table 9, 10, 11 and 12, I got a worse scenario when considering only s around of the annual cycles 252 and 300 when the range of the data sample is small, which makes me believe that the annual part is not yet outlined when I consider small sample sizes.

Table 6: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = \phi_m = \phi_a = 0.3$

	$\phi_w = \phi_m = \phi_a = 0.3$			
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	99.6	99.9	99.9	97.8
1000	100.0	100.0	100.0	100.0
2000	100.0	100.0	100.0	100.0
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 7: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = \phi_m = \phi_a = 0.2$

	$\phi_w = \phi_m = \phi_a = 0.2$			
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	98.1	99.4	98.9	83.0
1000	99.8	100.0	100.0	99.2
2000	100.0	100.0	100.0	100.0
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 8: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = \phi_m = \phi_a = 0.1$

$\phi_w = \phi_m = \phi_a = 0.1$				
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	70.6	79.1	76.2	51.6
1000	91.0	96.3	93.7	76.7
2000	99.8	100.0	99.5	95.6
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 9: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = 0.4, \phi_m = 0.3, \phi_a = -0.2$

$\phi_w = 0.4, \phi_m = 0.3, \phi_a = -0.2$				
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	98.2	99.8	99.8	81.3
1000	99.5	100.0	100.0	98.5
2000	99.8	100.0	100.0	99.9
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 10: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = 0.4, \phi_m = -0.3, \phi_a = 0.2$

$\phi_w = 0.4, \phi_m = -0.3, \phi_a = 0.2$				
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	99.5	99.9	99.8	83.2
1000	99.9	100.0	100.0	99.3
2000	100.0	100.0	100.0	100.0
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Table 11: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = -0.4, \phi_m = 0.3, \phi_a = 0.2$

$\phi_w = -0.4, \phi_m = 0.3, \phi_a = 0.2$				
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	100.0	100.0	100.0	83.5
1000	100.0	100.0	100.0	99.6
2000	100.0	100.0	100.0	99.9
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

3.2 Second Simulation Exercise: Testing for ACF

Table 12: Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles. Simulations for ACF when $\phi_w = 0.4, \phi_m = -0.3, \phi_a = -0.2$

	$\phi_w = 0.4, \phi_m = -0.3, \phi_a = -0.2$			
T	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{302}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$ $[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$	$[\rho_4, \rho_5]$ $[\rho_{20}, \rho_{26}]$	$[\rho_{250}, \rho_{254}]$ $[\rho_{297}, \rho_{301}]$
500	99.7	100.0	99.9	82.2
1000	99.9	100.0	100.0	99.1
2000	100.0	100.0	100.0	100.0
5000	100.0	100.0	100.0	100.0
10000	100.0	100.0	100.0	100.0

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

4 Discussion

After describing the simulations made in section 3, it is time to deepen knowledge on the topic further and somehow to complete and give a different perspective to the analysis previously done.

The tool used to generate and to do the entire simulation was in Python language, and part of the code that is used to compute the percentage of times that N series were correctly clustered can be found in Appendix B[3]

Let me first compare both methods, *fragmented-periodogram* and *fragmented-ACF*, using the lower and upper bounds that were used in Caiado, Crato, and Poncela (2020) to create a method that works for frequency-domain and time-domain, this method is showed in table 1 and 2 for the *fragmented-periodogram* and for the *fragmented-ACF*, this method is heuristically represented in table 4 and table 5. When comparing both cases, using the whole periodogram works better than using the entire ACF. However, since the intent is to reduce computational work, we are more interested in this study to have a promising approach for cutting data. Moreover, in this case, using these upper and lower boundaries approach represented in equation 2 and equation 3, fragmenting the ACF will give us the most success in terms of clustering data. If looking from a spectral data perspective, and as seen in Caiado, Crato, and Poncela (2020) paper, doing the smooth will still bring the best results since we reduce the variance of the data.

It is interesting to analyze in a more targeted way to our point of seasonality question. Since I know precisely the seasonality points I want to study, I found it interest to see what it would be like to test for specific boundaries around the s, highlighting the fact that the number of ordinates of periodogram and values of ACF will be the same, whatever the sample size T. The choice of these boundaries are presented in tables 3 and the remaining in appendix A.1, for the periodogram case, and tables 6, 7, 8, 9, 10, 11, 12 for the ACF case. For this approach, it is evident that it worked pretty well and better for the *fragmented-ACF*.

The main comment is that results are pretty aligned with those in (Caiado, Crato, and

[3]Part of code used to compute the periodogram and smoothed periodogram has the contribution of a GitHub page: Gates et al. (2021)

Poncela, 2020) regarding *fragmented-periodogram*, which gives us certain confidence for this new approach of the *fragmented-ACF*.

5 Conclusions and future work

Through the literature review, it is clear that nowadays, there are many different ways to assess time series similarities and dissimilarities, as well as consequently cluster them. It is also clear that different methods work well for some purposes and others for some other purposes.

One type of problem that has recently received much attention is the comparison and clustering of time series that are not correlated or may even come from different realms. One typical example is the analysis of heartbeat rhythms from different people in different parts of the globe and at different times. The comparison of the rhythms may give some valuable insight for medical diagnostic. Another example of this type of problem is the analysis of financial market assets that may be uncorrelated but have common stochastic features useful for general market analysis.

There are a few methods available for this the comparison and clustering of this type of problems. A very recent one is the fragmented periodogram approach, which has the additional characteristic of being computationally simple and thus appropriate to condensate information of large sets of long time series.

The main contribution of this work is the proposal and assessment of a similar method but in the time domain. This method uses the sample Auto Correlation Function (ACF) for each series under study, computes the ACF at specific lags only, and compares the series through the values obtained for the calculated lags.

A simulation exercise showed that this method works as well or even better than the fragmented periodogram. So, the proposed method is competitive, and it can be used for the type of problems described above.

A by-product of this work that is of interest by itself is the following. When assessing the method for choosing the window lengths for fragmenting the periodogram, simulations show that there is no consistency for a fixed number of periodogram ordinates. Thus, when using this method, it is important to adapt the window length to the number of observations of each time series.

5 Conclusions and future work

It should be clear from the previous sections that the fragmented ACF method has been tried on specific classes of time series for which the driving components are few and known. Furthermore, only stationary autoregressive models were studied.

This observation leads to a couple of suggestions for future work. Firstly, the type of series under comparison should be extended in order to include other types of models, namely nonstationary. Secondly, it will be useful to further study the chosen ACF components and whether they could be smoothed by a process similar to the spectral one. Thirdly, it would be interesting to find whether an automatic exploratory search could find the main driving components of models with unknown structure. This is a difficult task, but the research could start by computing regular spaced ACF lags and then narrowing the computation on the more relevant ones. Fourthly, it will be interesting to compare both time-domain and spectral-domain fragmented methods with non-fragmented procedures.

Appendices

A Appendix

A.1 Tables of periodogram simulation

Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles

	$\phi_w = \phi_m = \phi_a = .2$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	35.8	44.7	35.8	44.7	35.2	40.3	35.8	43.3
1000	50.6	53.6	41.9	46.0	34.8	34.9	40.7	49.3
2000	55.5	80.4	48.3	69.0	39.9	46.4	47.4	68.8
5000	55.1	85.0	40.6	54.6	41.6	50.4	34.5	46.6
10000	54.4	83.6	48.0	68.9	34.0	38.6	51.3	73.1

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles

	$\phi_w = .4, \phi_m = .3, \phi_a = -.2$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	56.4	77.6	56.4	77.6	39.1	48.9	58.2	78.1
1000	71.9	94.2	69.5	90.8	37.1	38.2	75.4	93.3
2000	62.4	90.3	59.3	82.8	45.9	52.1	57.9	82.0
5000	53.2	88.5	45.8	72.0	48.7	67.5	37.6	54.7
10000	60.6	90.9	47.8	73.8	49.3	71.6	37.0	45.8

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

A.1 Tables of periodogram simulation

Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles

	$\phi_w = .4, \phi_m = -.3, \phi_a = .2$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	40.2	57.7	40.2	57.7	36.4	48.4	33.6	51.6
1000	72.4	94.3	50.3	67.4	38.9	39.6	49.1	69.6
2000	58.4	86.0	50.6	71.3	43.1	53.7	48.2	67.1
5000	90.6	100.0	63.4	91.6	37.3	42.4	69.0	92.2
10000	73.5	96.5	35.9	46.8	32.8	35.4	36.5	48.1

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles

	$\phi_w = -.4, \phi_m = .3, \phi_a = .2$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	52.9	76.5	52.9	76.5	37.5	43.6	59.7	79.5
1000	49.3	62.8	42.3	53.7	34.1	34.2	41.7	59.8
2000	95.6	100.0	65.3	96.1	37.1	41.1	72.0	96.9
5000	78.7	99.2	47.2	73.2	36.6	42.2	50.3	74.9
10000	39.8	51.7	37.9	47.5	32.1	33.7	39.0	49.6

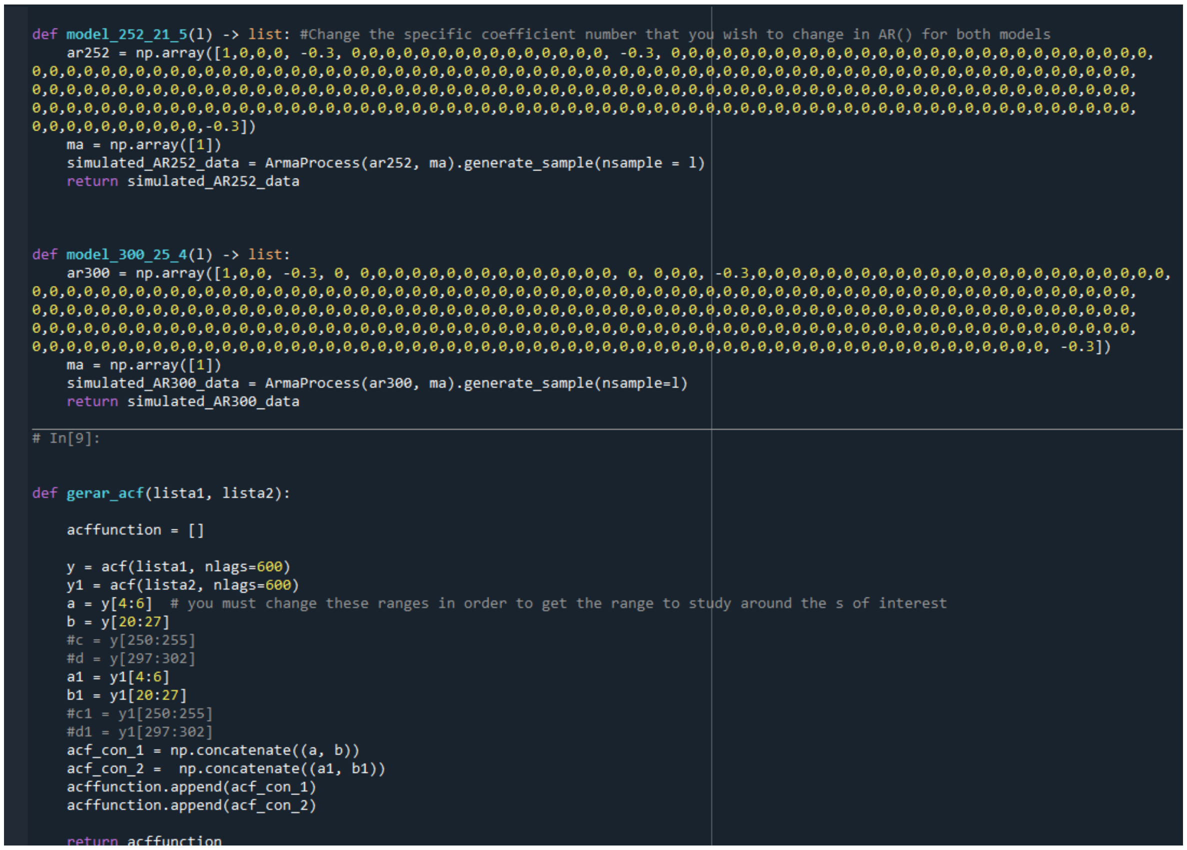
First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

Percentage of times that the $N = 4000$ series were correctly clustered when the 2 sets of series are different in the lags of weekly, monthly, and annual cycles

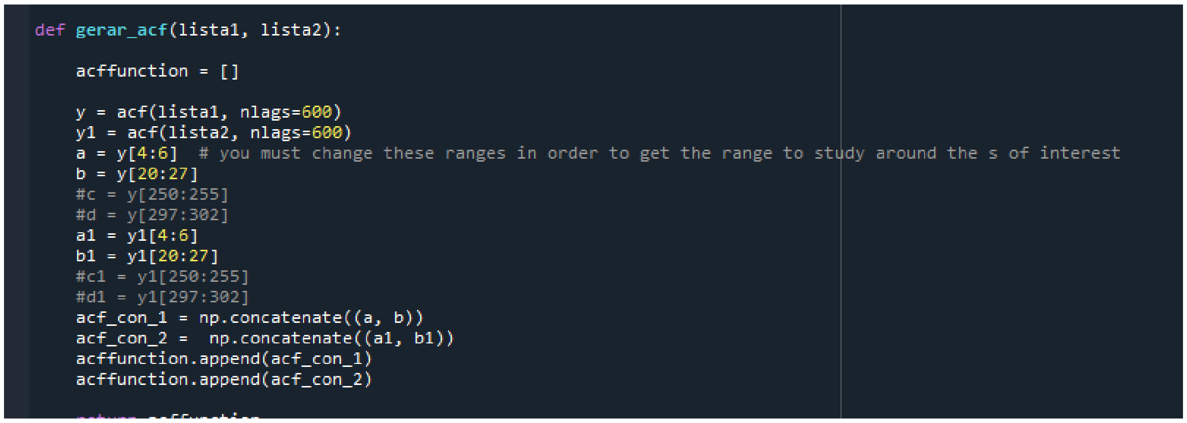
	$\phi_w = .4, \phi_m = -.3, \phi_a = -.2$							
	Frag	Smth.	Frag	Smth.	Frag	Smth.	Frag	Smth.
T	$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{302}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$ $[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$		$[I_4, I_5]$ $[I_{20}, I_{26}]$		$[I_{250}, I_{254}]$ $[I_{297}, I_{301}]$	
500	44.7	59.8	44.7	59.8	38.9	50.9	41.7	52.2
1000	71.6	93.3	51.1	64.8	43.1	45.3	50.4	69.0
2000	58.5	86.3	50.5	73.4	45.9	53.0	48.0	69.0
5000	89.9	99.8	47.1	74.3	37.0	43.4	45.9	75.5
10000	85.2	99.9	60.6	89.9	35.3	38.0	66.5	92.3

First group, lags = 5, 21 and 252; second group, lags = 4, 25 and 300

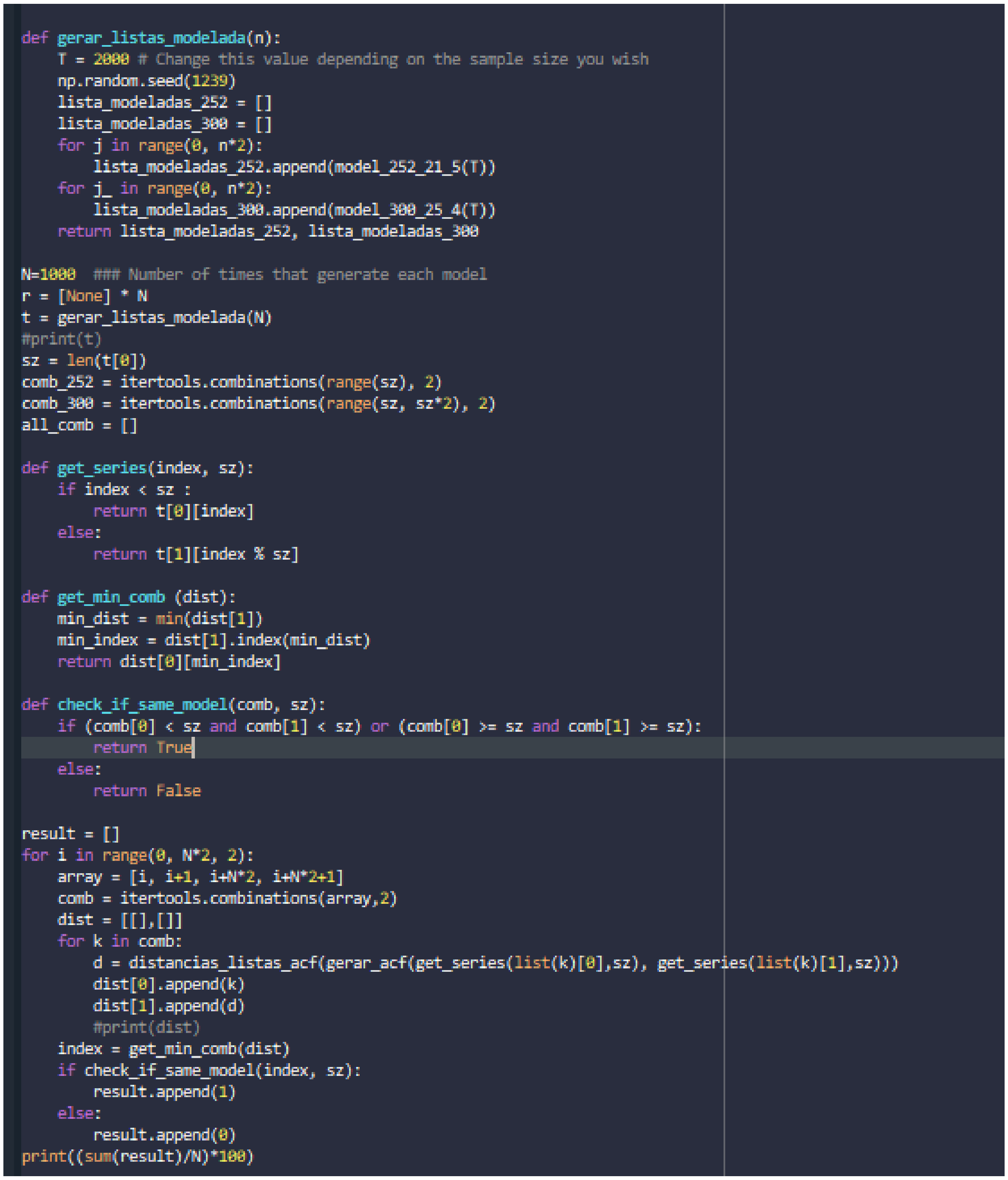
B Python Code



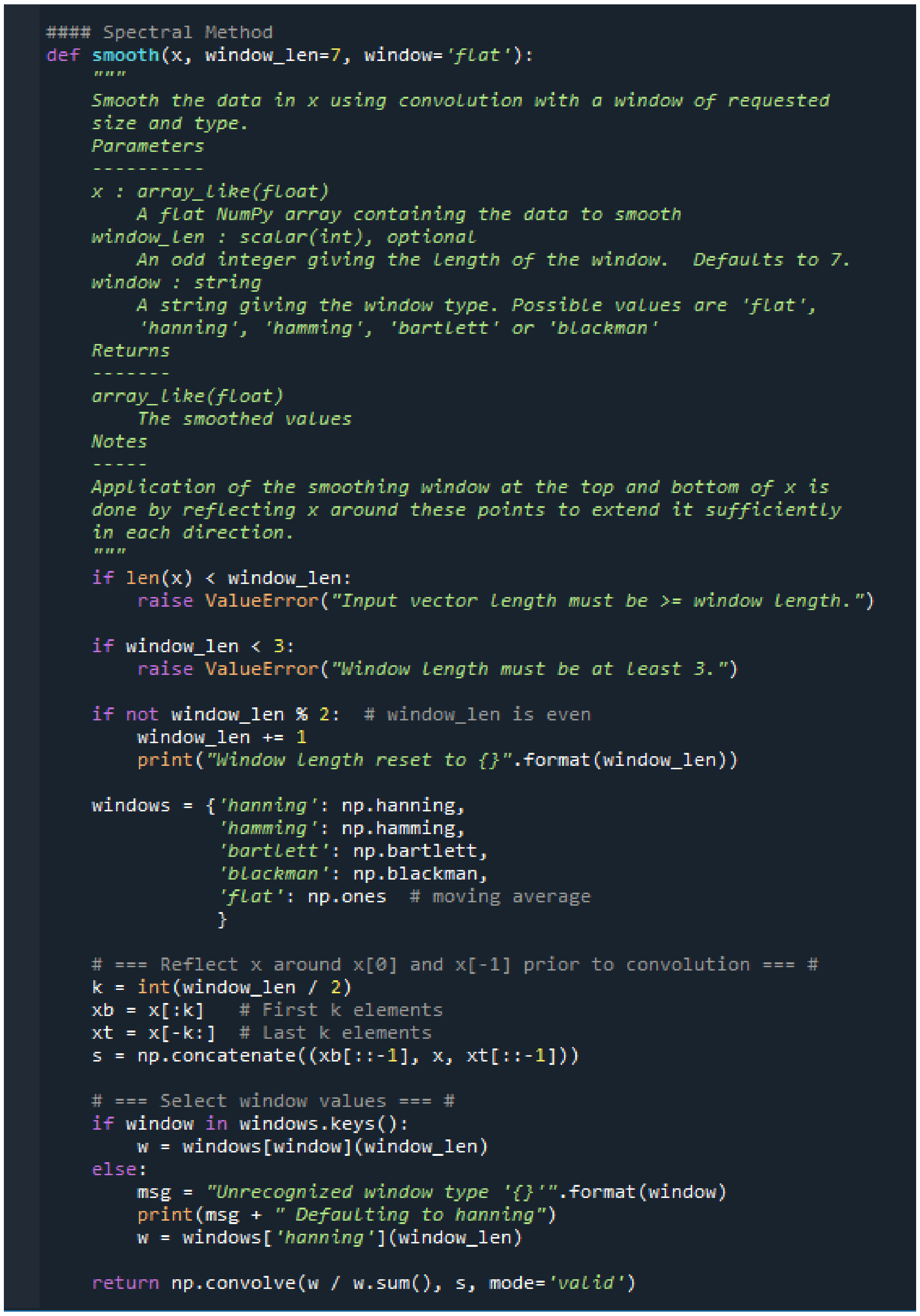
```
def model_252_21_5(l) -> list: #Change the specific coefficient number that you wish to change in AR() for both models
    ar252 = np.array([1,0,0,0, -0.3, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, -0.3, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,-0.3])
    ma = np.array([1])
    simulated_AR252_data = ArmaProcess(ar252, ma).generate_sample(nsample = 1)
    return simulated_AR252_data



def model_300_25_4(l) -> list:
    ar300 = np.array([1,0,0, -0.3, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0, 0,0,0, -0.3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, -0.3])
    ma = np.array([1])
    simulated_AR300_data = ArmaProcess(ar300, ma).generate_sample(nsample=1)
    return simulated_AR300_data

# In[9]:


def gerar_acf(lista1, lista2):

    acffunction = []

    y = acf(lista1, nlags=600)
    y1 = acf(lista2, nlags=600)
    a = y[4:6]  # you must change these ranges in order to get the range to study around the s of interest
    b = y[20:27]
    #c = y[250:255]
    #d = y[297:302]
    a1 = y1[4:6]
    b1 = y1[20:27]
    #c1 = y1[250:255]
    #d1 = y1[297:302]
    acf_con_1 = np.concatenate((a, b))
    acf_con_2 =  np.concatenate((a1, b1))
    acffunction.append(acf_con_1)
    acffunction.append(acf_con_2)

    return acffunction
```



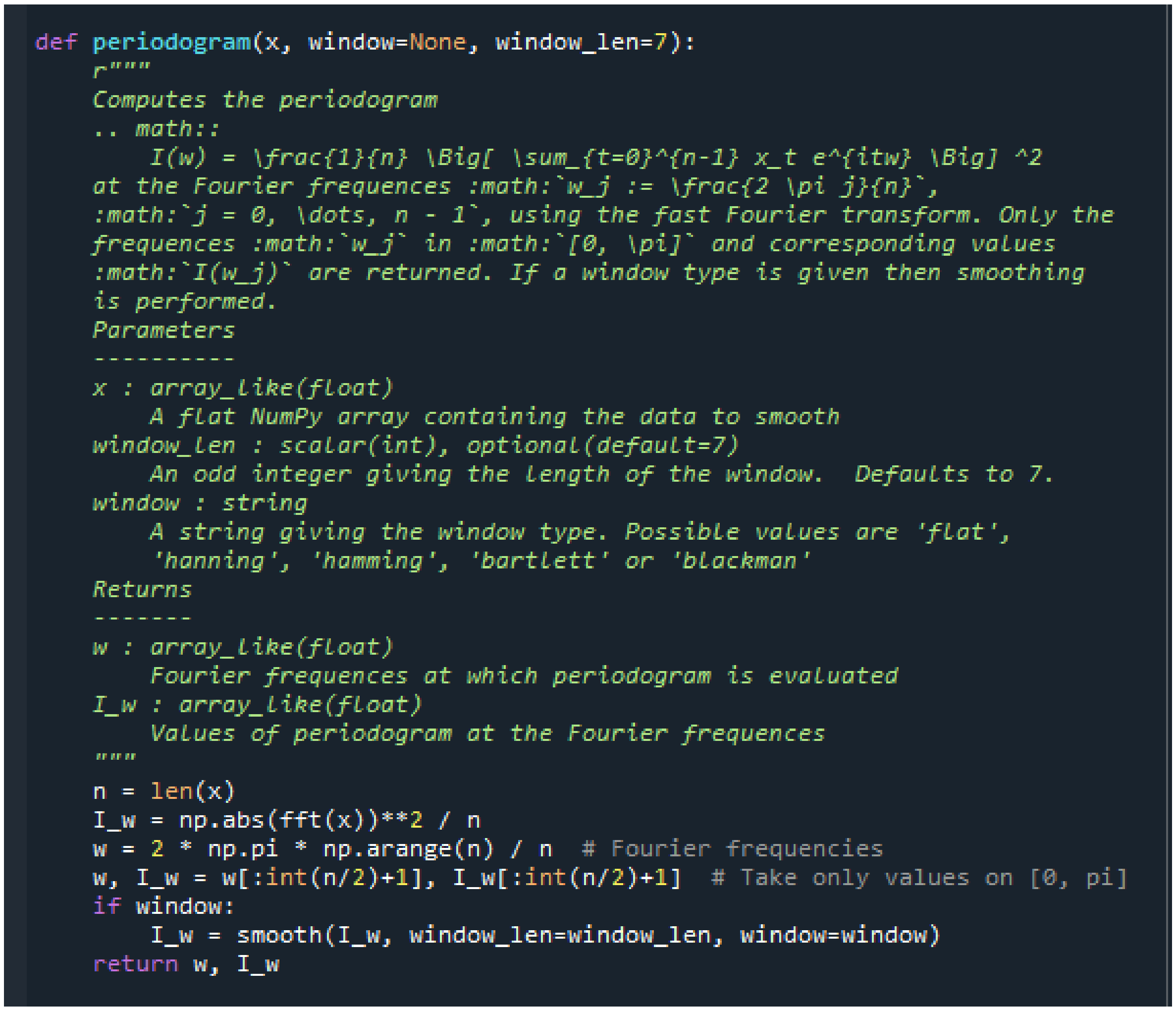
```
def gerar_acf(lista1, lista2):

    acffunction = []

    y = acf(lista1, nlags=600)
    y1 = acf(lista2, nlags=600)
    a = y[4:6]  # you must change these ranges in order to get the range to study around the s of interest
    b = y[20:27]
    #c = y[250:255]
    #d = y[297:302]
    a1 = y1[4:6]
    b1 = y1[20:27]
    #c1 = y1[250:255]
    #d1 = y1[297:302]
    acf_con_1 = np.concatenate((a, b))
    acf_con_2 =  np.concatenate((a1, b1))
    acffunction.append(acf_con_1)
    acffunction.append(acf_con_2)
```



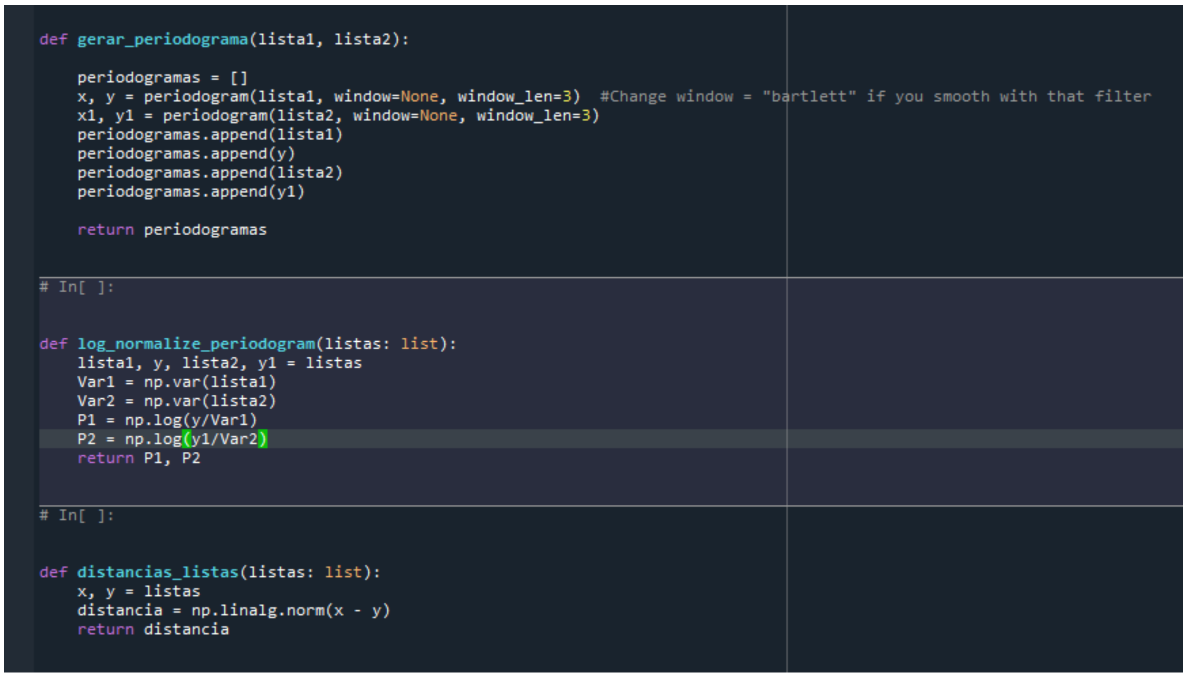
```python
def gerar_listas_modelada(n):
    T = 2000 # Change this value depending on the sample size you wish
    np.random.seed(1239)
    lista_modeladas_252 = []
    lista_modeladas_300 = []
    for j in range(0, n*2):
        lista_modeladas_252.append(model_252_21_5(T))
    for j_ in range(0, n*2):
        lista_modeladas_300.append(model_300_25_4(T))
    return lista_modeladas_252, lista_modeladas_300

N=1000  ### Number of times that generate each model
r = [None] * N
t = gerar_listas_modelada(N)
#print(t)
sz = len(t[0])
comb_252 = itertools.combinations(range(sz), 2)
comb_300 = itertools.combinations(range(sz, sz*2), 2)
all_comb = []

def get_series(index, sz):
    if index < sz :
        return t[0][index]
    else:
        return t[1][index % sz]

def get_min_comb (dist):
    min_dist = min(dist[1])
    min_index = dist[1].index(min_dist)
    return dist[0][min_index]

def check_if_same_model(comb, sz):
    if (comb[0] < sz and comb[1] < sz) or (comb[0] >= sz and comb[1] >= sz):
        return True
    else:
        return False

result = []
for i in range(0, N*2, 2):
    array = [i, i+1, i+N*2, i+N*2+1]
    comb = itertools.combinations(array,2)
    dist = [[],[]]
    for k in comb:
        d = distancias_listas_acf(gerar_acf(get_series(list(k)[0],sz), get_series(list(k)[1],sz)))
        dist[0].append(k)
        dist[1].append(d)
        #print(dist)
    index = get_min_comb(dist)
    if check_if_same_model(index, sz):
        result.append(1)
    else:
        result.append(0)
print((sum(result)/N)*100)
```



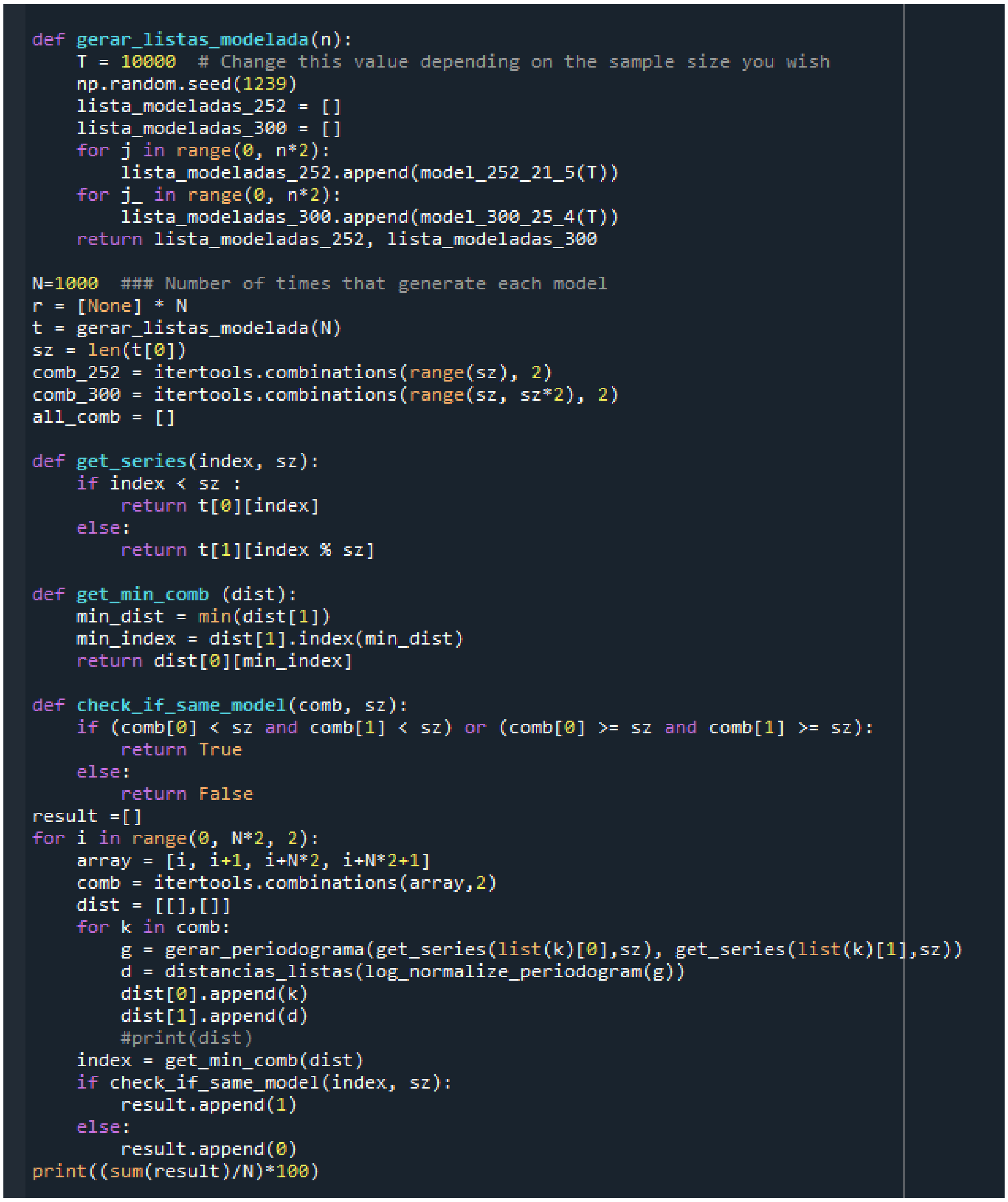
```python
#### Spectral Method
def smooth(x, window_len=7, window='flat'):
    """
    Smooth the data in x using convolution with a window of requested
    size and type.
    Parameters
    ----------
    x : array_like(float)
        A flat NumPy array containing the data to smooth
    window_len : scalar(int), optional
        An odd integer giving the length of the window.  Defaults to 7.
    window : string
        A string giving the window type. Possible values are 'flat',
        'hanning', 'hamming', 'bartlett' or 'blackman'
    Returns
    -------
    array_like(float)
        The smoothed values
    Notes
    -----
    Application of the smoothing window at the top and bottom of x is
    done by reflecting x around these points to extend it sufficiently
    in each direction.
    """
    if len(x) < window_len:
        raise ValueError("Input vector length must be >= window length.")

    if window_len < 3:
        raise ValueError("Window length must be at least 3.")

    if not window_len % 2:  # window_len is even
        window_len += 1
        print("Window length reset to {}".format(window_len))

    windows = {'hanning': np.hanning,
               'hamming': np.hamming,
               'bartlett': np.bartlett,
               'blackman': np.blackman,
               'flat': np.ones  # moving average
               }

    # === Reflect x around x[0] and x[-1] prior to convolution === #
    k = int(window_len / 2)
    xb = x[:k]   # First k elements
    xt = x[-k:]  # Last k elements
    s = np.concatenate((xb[::-1], x, xt[::-1]))

    # === Select window values === #
    if window in windows.keys():
        w = windows[window](window_len)
    else:
        msg = "Unrecognized window type '{}'".format(window)
        print(msg + " Defaulting to hanning")
        w = windows['hanning'](window_len)

    return np.convolve(w / w.sum(), s, mode='valid')
```



```python
def periodogram(x, window=None, window_len=7):
    r"""
    Computes the periodogram
    .. math::
        I(w) = \frac{1}{n} \Big[ \sum_{t=0}^{n-1} x_t e^{itw} \Big] ^2
    at the Fourier frequences :math:`w_j := \frac{2 \pi j}{n}`,
    :math:`j = 0, \dots, n - 1`, using the fast Fourier transform. Only the
    frequences :math:`w_j` in :math:`[0, \pi]` and corresponding values
    :math:`I(w_j)` are returned. If a window type is given then smoothing
    is performed.
    Parameters
    ----------
    x : array_like(float)
        A flat NumPy array containing the data to smooth
    window_len : scalar(int), optional(default=7)
        An odd integer giving the length of the window.  Defaults to 7.
    window : string
        A string giving the window type. Possible values are 'flat',
        'hanning', 'hamming', 'bartlett' or 'blackman'
    Returns
    -------
    w : array_like(float)
        Fourier frequences at which periodogram is evaluated
    I_w : array_like(float)
        Values of periodogram at the Fourier frequences
    """
    n = len(x)
    I_w = np.abs(fft(x))**2 / n
    w = 2 * np.pi * np.arange(n) / n  # Fourier frequencies
    w, I_w = w[:int(n/2)+1], I_w[:int(n/2)+1]  # Take only values on [0, pi]
    if window:
        I_w = smooth(I_w, window_len=window_len, window=window)
    return w, I_w
```



```
def gerar_periodograma(lista1, lista2):

    periodogramas = []
    x, y = periodogram(lista1, window=None, window_len=3)  #Change window = "bartlett" if you smooth with that filter
    x1, y1 = periodogram(lista2, window=None, window_len=3)
    periodogramas.append(lista1)
    periodogramas.append(y)
    periodogramas.append(lista2)
    periodogramas.append(y1)

    return periodogramas

# In[ ]:


def log_normalize_periodogram(listas: list):
    lista1, y, lista2, y1 = listas
    Var1 = np.var(lista1)
    Var2 = np.var(lista2)
    P1 = np.log(y/Var1)
    P2 = np.log(y1/Var2)
    return P1, P2


# In[ ]:


def distancias_listas(listas: list):
    x, y = listas
    distancia = np.linalg.norm(x - y)
    return distancia
```



```
def gerar_listas_modelada(n):
    T = 10000  # Change this value depending on the sample size you wish
    np.random.seed(1239)
    lista_modeladas_252 = []
    lista_modeladas_300 = []
    for j in range(0, n*2):
        lista_modeladas_252.append(model_252_21_5(T))
    for j_ in range(0, n*2):
        lista_modeladas_300.append(model_300_25_4(T))
    return lista_modeladas_252, lista_modeladas_300

N=1000  ### Number of times that generate each model
r = [None] * N
t = gerar_listas_modelada(N)
sz = len(t[0])
comb_252 = itertools.combinations(range(sz), 2)
comb_300 = itertools.combinations(range(sz, sz*2), 2)
all_comb = []

def get_series(index, sz):
    if index < sz :
        return t[0][index]
    else:
        return t[1][index % sz]

def get_min_comb (dist):
    min_dist = min(dist[1])
    min_index = dist[1].index(min_dist)
    return dist[0][min_index]

def check_if_same_model(comb, sz):
    if (comb[0] < sz and comb[1] < sz) or (comb[0] >= sz and comb[1] >= sz):
        return True
    else:
        return False
result =[]
for i in range(0, N*2, 2):
    array = [i, i+1, i+N*2, i+N*2+1]
    comb = itertools.combinations(array,2)
    dist = [[],[]]
    for k in comb:
        g = gerar_periodograma(get_series(list(k)[0],sz), get_series(list(k)[1],sz))
        d = distancias_listas(log_normalize_periodogram(g))
        dist[0].append(k)
        dist[1].append(d)
        #print(dist)
    index = get_min_comb(dist)
    if check_if_same_model(index, sz):
        result.append(1)
    else:
        result.append(0)
print((sum(result)/N)*100)
```

References

Agrawal, Rakesh, Christos Faloutsos, and Arun Swami (1993). "Efficient similarity search in sequence databases". In: *International conference on foundations of data organization and algorithms*. Springer, pp. 69–84.

Berndt, Donald J and James Clifford (1996). "Finding patterns in time series: a dynamic programming approach". In: *Advances in knowledge discovery and data mining*, pp. 229–248.

Bettini, Claudio et al. (1998). "Discovering frequent event patterns with multiple granularities in time sequences". In: *IEEE Transactions on Knowledge and Data Engineering* 10.2, pp. 222–237.

Caiado, Jorge, Nuno Crato, and Daniel Peña (2006). "A periodogram-based metric for time series classification". In: *Computational Statistics amp; Data Analysis*. URL: `https://www.sciencedirect.com/science/article/pii/S0167947305000770`.

Caiado, Jorge, Nuno Crato, and Pilar Poncela (2020). "A fragmented-periodogram approach for clustering big data time series". In: *Advances in Data Analysis and Classification* 14.1, pp. 117–146.

Chan, FK-P, AW-C Fu, and Clement Yu (2003). "Haar wavelets for efficient similarity search of time-series: with and without time warping". In: *IEEE Transactions on knowledge and data engineering* 15.3, pp. 686–705.

Chauhan, Sapna, Pridhi Arora, and Pawan Bhadana (2013). "Algorithm for semantic based similarity measure". In: *Int. J. Eng. Sci. Invent* 2.6, pp. 75–78.

Cuzzocrea, Alfredo (2020). *Multidimensional Clustering over Big Data: Models, Issues, Analysis, Emerging Trends*. Vienna, Austria. DOI: `10.1145/3400903.3409117`. URL: `https://doi.org/10.1145/3400903.3409117`.

Diggle, Peter J and Nicholas I Fisher (1991). "Nonparametric comparison of cumulative periodograms". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 40.3, pp. 423–434.

Fancoua, Craig L and Jose C Principe (1996). “A neighborhood map of competing one step predictors for piecewise segmentation and identification of time series”. In: *Proceedings of International Conference on Neural Networks (ICNN’96)*. Vol. 4. IEEE, pp. 1906–1911.

Galeano, Pedro and Daniel Peña (2001). “Multivariate analysis in vector time series”. In: pp. 383–404.

Gates, Tim, Natasha, and Chase Coleman (2021). *QuantEcon*. `https://github.com/QuantEcon/QuantEcon.py/blob/master/quantecon/estspec.py`.

Jancey, R. (1966). “Multidimensional group analysis”. In: *Australian Journal of Botany* 14, pp. 127–130.

Kirlić, Ajla and Muhedin Hadžić (Jan. 2017). “Big data and time series: A literature review paper”. In: *Univerzitetska misao - casopis za nauku, kulturu i umjetnost, Novi Pazar*, pp. 139–146. DOI: `10.5937/univmis1716139K`.

Laney, Doug et al. (2001). “3D data management: Controlling data volume, velocity and variety”. In: *META group research note* 6.70, p. 1.

Peña, Daniel and Pilar Poncela (2006). “Nonstationary dynamic factor analysis”. In: *Journal of Statistical Planning and Inference* 136.4, pp. 1237–1257.

Piccolo, Domenico (1990). “A distance measure for classifying ARIMA models”. In: *Journal of time series analysis* 11.2, pp. 153–164.

Wei, William WS (2006). “Time series analysis”. In: *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*.