UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

DOBLE GRAU DE MATEMÀTIQUES I ENGINYERIA INFORMÀTICA

Treball final de grau

Persistent Homology: Functional Summaries of Persistence Diagrams for Time Series Analysis

Autor: Manuel Lecha Sánchez

Directors:	Dr. Miquel Crusells, Dr. Josep Vives
Realitzat a:	Departament de Matemàtiques i Informàtica

Barcelona, 7 d'abril de 2023

Abstract

Topological data analysis (TDA) is a recently emerged field of study, a point of confluence between Algebraic Topology, Statistics and Computation Theory, born to develop a new set of tools capable of extracting qualitative and quantitative information from the data's underlying geometrical and topological structure. In these notes, we first present the theoretical foundations of the flagship tool of TDA, persistent homology. Later, we provide a framework that allows us to understand homological persistence from a statistical perspective. The latter comprises a set of maps called functional summaries, which map persistence diagrams -a common representation of persistence homology- to L-Lipschitz functions, a more convenient representation for data analysis. We present persistence landscapes, silhouettes, and a new functional summary candidate based on persistence entropy under this framework.

From this point on, we will focus our work on TDA for time series [31], and more specifically, financial time series [35][32][17]. First, we present the time-delay embedding and the sliding window approach, two methodologies that will allow us to transform time series into sequences of point clouds, which is essential for applying homological persistence tools.

Subsequently, mainly motivated by [1], an article that we could consider an extension of [17], we prove that the results reflected on the dependency relationship between persistence landscapes functional norms and variance-covariance for multivariate time series embedded via the sliding window method are valid for time series understanding them as a realization of a weakly stationary stochastic process, and assuming that the point clouds are obtained by means of time delay embedding. Furthermore, we assert that the validity of the results provided in [1] and the validity of our adaptation hold for silhouettes.

Moreover, we provide two distinct Python frameworks for classical and topological data analysis, which serves us to validate results. First, we provide a collection of tools to simulate time series from standard probability distributions and time series models such as AR(1), ARCH(1), GARCH(1,1). We present the necessary tools to embed, plot, and compute, topological summaries such persistence diagrams, persistence landscapes (and their corresponding L_p norms), silhouettes (and their corresponding L_p norms), persistence entropy, ES and NES functions. Analogous work is done regarding statistical and topological data analysis of stock index data.

Resumen

El análisis topológico de datos (ATD) es un campo de estudio de reciente aparición, un punto de confluencia entre la Topología Algebráica, la Estadística y la Teoría de la Computación, nacido con el objetivo de desarrollar un nuevo conjunto de herramientas capaces de extraer información cualitativa y cuantitativa de la estructura geométrica y topológica subyacente de los datos. Al inicio de estos apuntes proporcionamos los fundamentos teóricos de la herramienta insignia del ATD, la homología persistente. Posteriormente, proporcionamos un marco que nos permite entender la persistencia homolǵica desde una perspectiva estadística. Este último comprende un conjunto de mapas llamados resuúmenes funcionales, que mapean los diagramas de persistencia -una representación común de la persistencia homológica- con funciones L -Lipschitz, una representación que resulta más conveniente para el análisis de datos. Bajo este marco, presentamos los paisajes de persistencia, las siluetas y un nuevo mapa basado en la entropÃa de persistencia.

A partir de este momento, centraremos nuestro trabajo en ATD para series temporales [31], y más específicamente, series temporales financieras [35][32][17]. En primer lugar, presentamos el *time delay embedding* y el *sliding window method*, dos metodologías que nos permitirán transformar series de tiempo en secuencias de nubes de puntos, lo cual es esencial para aplicar herramientas de persistencia homológica.

Posteriormente, motivados principalmente por [1], un artículo que podríamos considerar una extensión de [17], comprobamos que los resultados que reflejan la relación de dependencia entre las normas funcionales de los paisajes de persistencia y la varianza-covarianza de las series de tiempo multivariantes tratadas con el *sliding window method* son también válidos para series de tiempo entendidas como realizaciones de procesos estocásticos débilmente estacionarios y asumiendo que las nubes de puntos asociadas se han obtenido mediante el *time delay embedding*. Por otro lado, afirmamos la validez de los resultados proporcionados en [1] y la validez de nuestra adaptación para las siluetas.

Además, proporcionamos dos programas en Python para el análisis de datos clásico y topológico, que nos servirán para comprobar resultados. En primer lugar, proporcionamos una colección de herramientas para simular series temporales a partir de distribuciones de probabilidad estándar y de modelos de series temporales como AR (1), ARCH (1,1), GARCH (1,1). Presentamos métodos para obtener nubes de puntos, visualizar y calcular diagramas de persistencia, paisajes de persistencia (y sus correspondientes normas L_p), siluetas (y sus correspondientes normas L_p), entropía de persistencia, funciones ES y NES. Se realiza un trabajo anaĺogo con respecto al análisis estadístico y topológico de datos recogido de índices bursat́iles.

Acknowledgements

En primer lugar, querría expresar mi más sincera gratitud hacia mis tutores, el Dr. Josep Vives y el Dr. Miquel Crusells-Girona, quienes han constituido una pilar fundamental para este trabajo. Agradecerles su entera disposición y generosidad por el tiempo dedicado.

Agradezco a mis familiares y amigos su apoyo y confianza.

En memoria de mi madre, esta es una promesa que le he cumplido, aun desconozco el significado de muchas otras.

Contents

2010 Mathematics Subject Classification. 55-01, 55-04, 60G10,62-01,62-04,91-08

Chapter 1

Introduction

Topological data analysis (TDA) is a recently emerged field of study, a point of confluence between Algebraic Topology, Statistics and Computation Theory, born to develop a new set of tools capable of extracting qualitative and quantitative information from the data's underlying geometrical and topological structure. The set of tools encompassed by TDA is fairly diverse, for example, we could find the Mapper algorithm [41], capable of generating a simplified topological graph given any set of points in $\mathbb{R}^d$ together with a map. However, our work will be entirely based on persistent homology, the flagship tool of TDA, first introduced by Edelsbrunner, Letscher and Zomorodian [14].

Persistent homology provides a framework to study the topological features of data. It has been successfully applied in a large number of problems in fields as diverse as bioinformatics, medicine, neuroscience, astrophysics, economics and finance [39][30][32][17][22][31].

Geometry, Algebraic Topology and more specifically, simplicial homology constitute the mathematical basis on which persistent homology is built. That is why our work introduces simplicial homology's natural domain, simplicial complexes, which can be understood as geometrical or abstract combinatorial objects. We will show how simplicial complexes induce a particular algebraic structure called chain complex, which leads to the definition of simplicial homology groups.

Computing persistence homology necessitates data to be fitted into a simplicial complex following a topological scheme. Thus, our next concern will be to introduce Čech complexes and Vietoris Rips complexes, the latter being fundamental to our further work. We will see that under certain assumptions, both constructions are homotopy equivalent to the underlying space from which the data is sampled, and hence, homology remains invariant.

However, the Vietoris Rips does not provide a unique construction. Under this scheme, a certain set of $k+1$ points forms a k-simplex if a ball of radius ϵ can surround them. Hence, the yielded simplicial complex intrinsically depends on a parameter and, any alteration of it could entirely modify the resulting topological structure.

Persistence homology, which can be understood as the standard homology of a particular graded module over a polynomial ring, provides a topological invariant encompassing the homology groups induced by all these possible perturbations of ϵ. The latter will allow us to know which topological features persist during a wide range of parameter variations, i.e., the significant topological features. We will see how persistence homology can be captured in a particular set of points in $\mathbb{R}^2$ known as persistence diagram, where each point represents the birth (x-axis) and death (y-axis) of a homology generator.

TDA works with data, and hence, we should not ignore the fundamental aspects of classical data analysis. Under certain assumptions, a probability space over persistence diagrams can be defined. However, basic operations, such as expectations, become problematic. Thus, we will introduce functional summaries, which map persistence diagrams to a collection of functions that, under the condition of being L-Lipschitz, provide a better framework for statistics, where, for instance, pointwise convergence holds. We present persistence landscapes and silhouettes under this framework, which, by construction, satisfy the condition of being one-Lipschitz functions. Moreover, we introduce a new functional summary candidate based on persistence entropy, which satisfies the latter.

The work that follows is motivated by recent articles in TDA for time series analysis and financial time series such as [17][31][35][32]. Notably, in [17], M. Guidea and Y. Katz develop an EWS algorithm based on persistence landscapes L_p norms. To obtain the sequence of point clouds, they apply the sliding window method to a 4-dimensional signal consisting of four 1-dimensional time series of stock indexes. However, the latter requires multiple time series to be performed. When persistence homology has to be computed on a single time series, as we will do, the standard approach relies on Takens' embedding [43] (embedding process to point cloud) and the sliding window method (obtain a partition or sequence of point clouds) [31][35][32][18]. Hence, we presumed to be necessary introducing both.

At this point, our work focuses on an article provided by Josep Vives, of which he is a co-author [1], that can be considered as a continuation of [17]. In fact, we prove that the results reflected on the dependency relationship between persistence landscapes functional norms and variance-covariance in multivariate time series embedded via the sliding window method are valid for time series understanding them as a realization of a weakly stationary stochastic process, and assuming that the point clouds are obtained through the time-delay embedding.

We use the AR(1) model to show that dependency. Besides, we assert the validity of the previous results for silhouettes.

Finally, we provide two distinct Python frameworks for classical and topological data analysis, which serves us to validate results.

On the one hand, in FinTDA.ipybn A.1, we have developed a class (IndexTDA) with which several empirical results can be easily computed from stock indexes in a selectable date range. One can apply Takens' Embedding and the sliding window method to compute persistence landscapes L_p norms, persistence entropy and other useful values such as the autocorrelation function, kurtosis, standard deviation, for each of the point clouds.

On the other hand, in EmpTDA.ipynb B.1, first, we provide a bunch of methods to simulate time series using stochastic models, such as AR(1), ARCH(1) and GARCH(1,1) and standard distributions. Moreover, we dispense the tools to compute Takens' embedding, persistence diagrams, persistence landscape, power weighted silhouettes, persistence entropy, ES and NES functions.

Chapter 2

Simplicial Homology

2.1 Simplicial Complexes

In this section, we introduce simplicial complexes, which are the natural domain of definition of simplicial homology. Those mathematical objects can be understood either as geometric objects admitting a topology or as abstract combinatorial structures. Both traits make them play a core role in computational topology.

2.1.1 Geometric simplicial complex

Given a set $\mathbb{X}$ lying on an Euclidean space, we say that the convex hull of $\mathbb{X}$ is the minimal convex set containing $\mathbb{X}$. Let $x_0, ..., x_k$ be points on a Euclidean space $\mathbb{R}^K$, then the convex hull of that set of points can be understood as the set of their possible convex combinations.

$$\Delta(x_0, ..., x_k) = \{\sum_{i=0}^{k} \lambda_i x_i | \sum_{i=0}^{k} \lambda_i = 1, \lambda_i \geq 0\,, \forall i\} \tag{2.1}$$

A **k-geometric simplex** is said to be the convex hull of a set of k + 1 affinely independent points. We say that a **face** from such a k-geometric simplex is the convex hull of any subset of the set of points which its convex hull defines the k-geometric simplex. The usual nomenclature calls vertices to the 0-dimensional geometric simplices, edges to the 1-simplices, triangles to the 2-simplices and tetrahedra to the 3-simplices.

It will be very important for the development of our work, being able to build structures made of glued complexes. Intuitively, a set composed of vertices, edges, triangles, and their higher analogues.

Definition 2.1. *A* ***geometric simplicial complex K*** *is a set of simplices satisfying the following:*

- *Every face of a simplex from K is in K.*
- *The pairwise intersection of simplices of K is either empty or a common face of each of them.*

For further sections, it will be more convenient to get rid of the geometry and define the purely combinatorial counterpart to the geometric simplicial complex, known as an abstract simplicial complex. This process is called abstraction.

Abstraction

Given a geometrical simplicial complex K with a set of vertex $Vert(K) = \{x_0, ..., x_n\} \in \mathbb{R}^K$, we consider $V = \{x_0, ..., x_n\}$ as the set of abstract vertices. Additionally, for each $\sigma \in K$, $\sigma = \{x_{i_0}, ..., x_{i_k}\}$ consider $\check{\sigma} = \{i_0, ..., i_k\}$ the abstract simplex. As the reader may note by reading the subsection below, under this construction, we obtain an abstract simplicial complex $\check{K}$. In fact, we say $\check{K}$ to be an abstraction of K.

As mentioned above, instead of delving into this subsection, we will deepen more in the abstract simplicial complex concept due to its convenience. The following definitions are combinatorial equivalences to geometric notions related to geometric simplicial complexes.

2.1.2 Abstract simplicial complex

Definition 2.2. *An* ***abstract simplicial complex*** *is a pair (V, S), where V is a finite set, and S is a collection of non-empty subsets of V such that for all element $\sigma \in S$, $\tau \subseteq \sigma$ implies $\tau \in S$.*

The element σ of S is called **simplex** of S, and it is said to be a **k-simplex**, or equivalently, have dimension k, if $|\sigma| = k + 1$, i.e, the number of elements of σ is k+1. S_k denotes the set of k-simplices of S. Sometimes, we will call S_k as the k-skeleton of the abstract simplicial complex. The latter allows us to define the simplicial complex's dimension as the largest positive integer k such that S_k is not an empty set. To add some more common notation, we will say that V elements are usually called vertices. Observe that we shall not make any distinction between $v \in V$ and the 0-simplex $\{v\} \in S$.

Definition 2.3. *Given an abstract simplicial complex K, L is said to be a subcomplex of K if it is an abstract simplicial complex satisfying $L \subseteq K$*

Due to homology purposes, it will be crucial to keep track of the order of simplex's vertices. Hence, it is necessary to introduce the notion of an **ordered** abstract simplicial complex in this subsection.

Definition 2.4. *An abstract simplicial complex $K = (V, S)$ is said to be **ordered** if the set V is totally ordered.*

In such case, we will typically choose a bijection $\phi : V \to \{1, 2, ..N\}$ and denote the k-simplices as $\sigma = [v_0, ..., v_k]$ in such a way that $v_0 \leq ... \leq v_k$. Unless otherwise specified, whenever we talk about simplicial complexes, we will be referring to ordered abstract simplicial complexes.

Geometric realization

In the same way we have said that we can obtain an abstraction $\check{K}$ from a geometric simplicial complex K, we can contrariwise associate a topological space $|(V, S)|$ to an abstract simplicial complex and obtain what is called its **geometric realization**. This space can be defined as the subspace of $\mathbb{R}^N$ defined as $|(V, S)| = \bigcup\limits_{\sigma \in S} \Delta(\sigma)$ where ϕ is the bijection above, and $\Delta(\sigma)$ the convex hull of the set $\{e_{\phi(v)}\}_{v\in\sigma}$, each e_i representing the ith standard basis vector, i.e., a N-dimensional vector with a 1 in the ith position and 0 otherwise $(0, ..., 1_i, ..., 0)$.

2.1.3 Simplicial maps and triangulation of topological spaces

Given K, L abstract simplicial complexes, a vertex map is said to be a function $f : V_K \to V_L$ sending every vertex $v \in V_K$ to a vertex $f(v) \in V_L$. In our work, we need maps preserving the structure of the space.

Definition 2.5. *Given K, L abstract simplicial complexes, a map f is called a **simplicial map** if it is a vertex map sending simplices to simplices; that is, being $[v_0, ..., v_k]$ a simplex of K, implies $[f(v_0), ..., f(v_k)]$ being a simplex of L. Notice that since f does not need to be injective, it could appear redundancy in the vertexes of the simplex $[f(v_0), ..., f(v_k)]$, and hence, have a lower dimension than $[v_0, ..., v_k]$.*

Then, we can extend the notion of a simplicial map f to a continuous map between $|K|$ and $|L|$. Since every $x \in |K|$ belongs to the interior of one unique simplex, according to 2.1 the latter can done by setting $f(x) = \sum_{i=0}^{d} \lambda_i f(x_i)$ where $\sigma = \{x_0, ..., x_d\}$ is such that $x \in \sigma^\circ$. Simplicial complexes together with simplicial maps offer a piecewise linear counterpart to topological spaces and continuous maps.

There is a class of topological spaces - *triangulable spaces* - which can be represented by simplicial complexes up to homeomorphism.

Definition 2.6. *A* ***triangulation*** *of a topological space X is a homeomorphism $h : |K| \to X$ where K is an abstract simplicial complex.*

A topological space X is said to be **triangulable** if there exists an abstract simplicial complex K such that $|K| \cong X$. Therefore, we can study the topological invariants of any triangulable topological from a homeomorphic simplicial complex, which gives a discrete representation of it.

2.2 Chain complexes and Simplicial Homology Groups

The primary purpose now is to define the simplicial homology groups for a given abstract simplicial complex K. If we can intuitively understand abstract simplicial complexes as a generalization of graphs, their corresponding homology groups can be understood as higher dimension analogues to graph connectivity. They provide a mathematical language to capture k-dimensional holes by studying what surrounds them. Additionally, they are topological invariants; that is, two homeomorphic topological spaces have isomorphic homology groups.

2.2.1 Chain complexes

Free R-modules over $\mathbf{S_n}$

First of all, we introduce the concept of a module over a ring, which intuitively corresponds to a generalization of the notion of a vector space, wherein the corresponding scalars are allowed to lie in an arbitrary ring.

Definition 2.7. *Given a ring R, a* ***left R-module M*** *over the ring R, denoted by $_{\mathbf{R}}\mathbf{M}$, consists of an abelian group $(M, +)$ and an operation $R \times M \to M$ (called scalar multiplication, usually just written by juxtaposition, i.e. as rx for r in R and x in M) such that for all r,s in R, x,y in M the following holds*

- $r(x + y) = rx + ry$
- $(r + s)x = rx + rs$
- $(rs)x = r(sx)$
- $1_R x = x$ *if R has a multiplicative identity* 1_R.

A right R-module M_R is similarly defined with the ring just acting on the right. Observe that if R is commutative, left and right R-modules are the same, and hence, we call them R-modules.

Remark 2.8. If K is a field, the concept of a vector space over K and the concept of a K-module are identical.

Remark 2.9. Every abelian group is a is a module over the ring of integers $\mathbb{Z}$ in a unique way. For $n \geq 0$, let $nx = x + x + ... + x$ (n summands), $0x = 0$ and $(-n)x = -nx$.

Definition 2.10. *Given a ring R and a R-module M, the set $E \subsetneqq M$ is a* ***basis for M*** *if:*

- *For all $m \in M$, $m = \sum_{i=0}^{|E|} r_i e_i$ for all $r_i \in R$, $e_i \in E$, i.e, E is a generating set.*
- *For all $E' \subseteq E$, $E' = \{e_{i_1}, ..., e_{i_n}\}$, if $\sum_{j=1}^{n} r_j e_{i_j} = 0_M$, it implies that $r_1 = ... = r_n = 0_R$, i.e, E is linearly independent.*

Definition 2.11. *It is said that a module M with a basis $E \subseteq M$ is a* ***free module****.*

Remark 2.12. Directly from the definition of a free module, we can observe that for given a set S and a ring R, we can induce an R-module with basis S called free module on S.

Definition 2.13. *Free modules over the ring of integers $\mathbb{Z}$ are called* ***free abelian groups****.*

Recall that a group G is said to be a free group if there exists a generating set X of G such that every non-empty reduced group word in X defines a non-trivial element of G. Hence, the reader must not confuse free abelian groups with free groups.

To illustrate, in free groups ab must be different from ba if a and b are different elements of the basis, however, in free abelian groups, ab must be equal to ba. Free abelian groups are abelian groups but not free groups except in two specific cases, when the group is the trivial group or when it has just one element in the basis, i.e., the infinite cyclic group.

Recall that given an ordered abstract simplicial complex $K = (V, S)$ we have defined S_n as its n-simplices set. Hence, observe that, accordingly to 2.12 , given an ordered abstract simplicial complex $K = (V, S)$ and a ring R, we can construct the free module over S_n. The latter will be denoted as $C_n(K, R)$. Henceforth, we will assume R to be either the integral domain $\mathbb{Z}$ or a field $\mathbb{F}$.

In this context, elements of $C_n(K, R)$ are called **n-chains**, and as a consequence of the imposed total order on K, each $c \in C_n(K)$ can be written as a formal sum

$$c = \sum_{\sigma \in S_n} n_\sigma \sigma, \ \ n_\sigma \in R$$

where every σ is a oriented n-simplex. The opposite orientation of a certain n-simplex in the chain will be denoted with its corresponding opposite sign coefficient of the chain .

Remark 2.14. Note that if $R = \mathbb{Z}$, $C_n(K, \mathbb{Z})$ is the free abelian group over S_n. In such case, to ease notation we may refer to $C_n(K, \mathbb{Z})$ as $C_n(K)$.

Boundary operator

Given a a n-simplex $\sigma = [v_0, .., v_n]$, its boundary corresponds to the n (n-1)-simplices or faces $[v_0, ..., \hat{v}_i, ..., v_n]$, meaning by ^ that the corresponding vertex is suppressed. In terms of chains we will denote it by $\sum\limits_{i=0}^{n} (-1)^i [v_0, ..., \hat{v}_i, ..., v_n]$, using the signs so as to keep faces of the simplex well-oriented. We now define the following:

Definition 2.15. *The **n-boundary operator** or **n-boundary homeomorphism** is* $\delta_n : C_n(K, R) \to C_{n-1}(K, R)$ *where* $\delta_n([v_0, .., v_n]) := \sum\limits_{i=0}^{n} (-1)^i [v_0, ..., \hat{v}_i, ..., v_n]$.

Lemma 2.16. $\mathrm{Im}(\delta_{n-1} \circ \delta_n) = \{0\}$

Proof. By definition we have that $\delta_n([v_0, .., v_n]) = \sum\limits_{i=0}^{n} (-1)^i [v_0, ..., \hat{v}_i, ..., v_n]$, then

$$\begin{aligned}\delta_{n-1}(\delta_n([v_0, .., v_n])) = \sum_{i=0}^{n} (-1)^i \delta_{n-1}([v_0, ..., \hat{v}_i, ..., v_n]) = \\ \sum_{j<i} (-1)^i (-1)^j [v_0, ..., \hat{v}_j, ..., \hat{v}_i, ..., v_n] \ + \\ \sum_{j>i} (-1)^i (-1)^{j-1} [v_0, ..., \hat{v}_i, ..., \hat{v}_j, ..., v_n].\end{aligned}$$

Now just notice that the elements of these two summations cancel in pairs by switching i and j, i.e, one becomes the negative of the other. □

Chain complex

Definition 2.17. *A **chain complex** (A_*, d_*) is an algebraic structure consisting of a sequence of abelian groups or modules $A_n, A_{n-1}, ..., A_1, A_0$ which are connected by boundary operators $d_n : A_n \to A_{n-1}$ satisfying* $\mathrm{Im}(d_{n-1} \circ d_n) = \{0\}$.

Thus, every ordered abstract simplicial complex determines a chain complex $(C_*(K, R), \delta_*)$ known as **simplicial chain complex**. Algebraically speaking, a simplicial chain complex $(C_*(K, R), \delta_*)$ represents the following situation:

$$...C_{n+2} \xrightarrow{\delta_{n+2}} C_{n+1} \xrightarrow{\delta_{n+1}} C_n \xrightarrow{\delta_n} ... \xrightarrow{\delta_2} C_1 \xrightarrow{\delta_0} C_0 \xrightarrow{0} 0$$

2.2.2 Simplicial homology groups and decomposition

Homology groups

Using the boundary operator we can define the following subgroups for each $C_n(K,R)$: the **cycle group** $\mathbf{Z_n(K,R)} = \text{Ker}(\delta_n)$, which is the group on n-cycles, and the **boundary group** $\mathbf{B_n(K,R)} = \text{Im}(\delta_{n+1})$, which is the group of n-boundaries. Notice that from Lemma 2.16 we get $B_n(K,R) \subseteq Z_n(K,R) \subseteq C_n(K,R)$.

Definition 2.18. *Given an abstract simplicial complex K, the* $\mathbf{n^{th}}$ ***simplicial homology group*** *over R, is the quotient group* $H_n(K,R) = H_n((C_*(K,R),\delta_*)) = Z_n(K,R)/B_n(K,R)$.

Decomposition

Decomposition over the integral domain $(\mathbf{Z})$ On the one hand, as stated previously, when coefficients are in the integral domain $\mathbb{Z}$, $C_n(K,\mathbb{Z})$, or also denoted as $C_n(K)$, is the free abelian group over S_n. Hence, since Richard Dedekind [21] proved that every subgroup of a free abelian group is a free abelian group itself, we have that $B_n(K,\mathbb{Z})$ and $Z_n(K,\mathbb{Z})$ are free abelian too.

However, the quotient of free $\mathbb{Z}$-modules does not necessarily have to be free. To illustrate, suppose $n > 1$, then $\mathbb{Z}$ and n$\mathbb{Z}$ are both free abelian groups, but since there is no non-empty subset which is linearly independent over $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$ is not a free $\mathbb{Z}$-module. Note that for all $x \in \mathbb{Z}/n\mathbb{Z}\ nx = 0$.

Nevertheless, since every quotient of a finitely generated abelian group is finitely generated abelian too, such homology groups are finitely generated abelian groups. Thus, in virtue of **the fundamental theorem of finitely generated abelian groups** [p. 175][42] we have the following decomposition:

$$H_n(K,\mathbb{Z}) = \mathbb{Z}^{\beta_n} \bigoplus_{\text{q prime},d_i>0} (\mathbb{Z}_{q_n^{d_1}} \oplus ... \oplus \mathbb{Z}_{q_n^{d_t}})$$

The first part of the sum is called the free part and $\beta_n \geq 0$ is called the **nth Betti number** of K. Note that $q_n^{d_i}$ are powers of not necessary distinct prime numbers. Those prime powers are said to be the torsion coefficients.

Decomposition over a field On the other hand, if R is a field $\mathbb{F}$, all the groups mentioned above became $\mathbb{F}$-vector spaces, and as a result, we have the following decomposition:

$$H_n(K,\mathbb{F}) \cong \mathbb{F}^{\beta_n}$$

In like manner as above $\beta_n \geq 0$ is called the nth Betti number of K. Note that in such case, $H_n(K,\mathbb{F})$ is torsion-free. Thus, the homology groups with coefficients in a field are fully determined by the rank β_n.

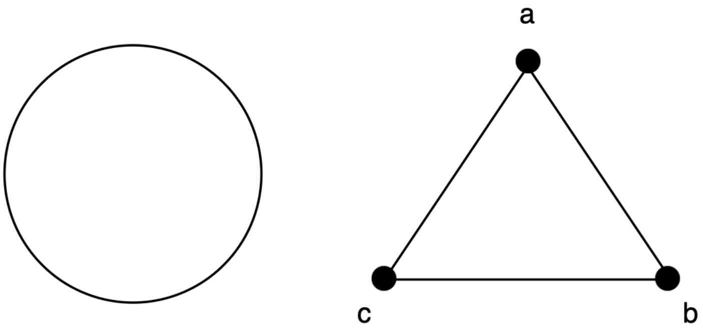
The **universal coefficient theorem for homology** [20] establishes the relationship between is homology over the integers and homology with field coefficients. One implication is that the latter is deductible from the first one. However, homology groups over a field, offer some computational advantages, and as we will see, better decomposition theorems for persistence.

A little simplicial homology computation example

In this subsection, we will show in two different manners, an approach to compute the simplicial homology of S^1. It will be very illustrative for our work to provide an example of a simple computation of the H_0 and H_1 groups.

Example 2.19. Suppose we want to compute the simplicial homology of S^1.

Figure 2.1: (Left to right) S^1 and a triangulation of S^1



First of all, we should observe that given the abstract simplicial complex $K = (V, S)$ such that $V = \{a, b, c\}$ and $S = \{[a], [b], [c], [a, b], [b, c], [a, c]\}$, its geometrical realization $|K| \cong S^1$. Thus, we have that S^1 is a triangulable space, and we automatically can compute its simplicial homology, by computing the simplicial homology of K. In this example we are going to assume all the computations over the coefficients of the field $\mathbb{Z}/2\mathbb{Z}$. Therefore, as mentioned above, n-chains become vector spaces and $H_0 = (\mathbb{Z}/2\mathbb{Z})^{\beta_0}$and $H_1 = (\mathbb{Z}/2\mathbb{Z})^{\beta_1}$.

To begin with, we are going to compute H_0 and H_1 in the "hand-craft" way:

Observe that in this case we have the following situation

$$C_2(K, \mathbb{Z}/2\mathbb{Z}) \xrightarrow{\delta_2} C_1(K, \mathbb{Z}/2\mathbb{Z}) \xrightarrow{\delta_1} C_0(K, \mathbb{Z}/2\mathbb{Z}) \xrightarrow{\delta_0=0} 0$$

where

- $C_2(K, \mathbb{Z}/2\mathbb{Z}) = 0$
- $C_1(K, \mathbb{Z}/2\mathbb{Z}) = \mathbb{Z}/2\mathbb{Z}[a, b] \oplus \mathbb{Z}/2\mathbb{Z}[b, c] \oplus \mathbb{Z}/2\mathbb{Z}[a, c] \cong (\mathbb{Z}/2\mathbb{Z})^3$

- $C_0(K, \mathbb{Z}/2\mathbb{Z}) = \mathbb{Z}/2\mathbb{Z}[a] \oplus \mathbb{Z}/2\mathbb{Z}[b] \oplus \mathbb{Z}/2\mathbb{Z}[c] \cong (\mathbb{Z}/2\mathbb{Z})^3$

To compute $H_0(K, \mathbb{Z}/2\mathbb{Z})$, we need $Z_0(K, \mathbb{Z}/2\mathbb{Z}) = \mathrm{Ker}(\delta_0)$ and $B_0(K, \mathbb{Z}/2\mathbb{Z}) = \mathrm{Im}(\delta_1)$. Since $\delta_0 = 0$, $\mathrm{Ker}(\delta_0) = C_0(K, \mathbb{Z}/2\mathbb{Z}) \cong (\mathbb{Z}/2\mathbb{Z})^3$. On the other hand, $\mathrm{Im}(\delta_1) =< [b] - [a], [c] - [b], [c] - [a] >$, since $[c] - [a] = ([b] - [a]) + ([c] - [b])$, we have that $\mathrm{Im}(\delta_1) =< [b] - [a], [c] - [b] >$. Thus, $\mathrm{Im}(\delta_1) \cong (\mathbb{Z}/2\mathbb{Z})^2$. Therefore, $H_0(K, \mathbb{Z}/2\mathbb{Z}) = {}^{(\mathbb{Z}/2\mathbb{Z})^3}/_{(\mathbb{Z}/2\mathbb{Z})^2} \cong \mathbb{Z}/2\mathbb{Z}$. Intuitively, this means that S^1 has one connected component. Note that in the case of $H_1(K, \mathbb{Z}/2\mathbb{Z})$, since we do not have any 2-simplex, $\mathrm{Im}(\delta_2) = \{0\}$ and therefore, $H_1(K, \mathbb{Z}/2\mathbb{Z}) = \mathrm{Ker}(\delta_1)$. Hence, we need to find the 1-cycles of our complex, $\delta_1(C_1(K, \mathbb{Z}/2\mathbb{Z})) = \alpha_1([b] - [a]) + \alpha_2([c] - [b]) + \alpha_3([c] - [a])$ with $\alpha_1, \alpha_1, \alpha_3 \in \mathbb{Z}/2\mathbb{Z}$. Therefore, $\delta_1(C_1(K, \mathbb{Z}/2\mathbb{Z})) = [a](-\alpha_1 - \alpha_3) + [b](\alpha_1 - \alpha_2) + [c](\alpha_2 - \alpha_3) = 0$ if and only if $\alpha_1 = \alpha_2 = \alpha_3$ and $\mathrm{Ker}(\delta_1) \cong \mathbb{Z}/2\mathbb{Z}$, then $H_1(K, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}$. The latter intuitively means that S^1 has 1D-loop.

As mentioned, we are going to compute the latter with another approach:

Observe that in this case, we have just taken care of the dimension of the homology group, i.e, the Betti number, which can be understood as $\beta_k = \dim(\mathrm{Ker}(\delta_k)) - rank(\delta_{k+1})$. Moreover, recall the following fundamental theorem of linear algebra for finite vector spaces, that states that for E a finite-dimensional vector space and u a linear application from E to some other vector space, we have that $dim(E) = rank(u) + \dim(\mathrm{Ker}(u))$. Therefore, we can turn this into finding the rank of δ_k (which will also give us the dimension of its kernel), which can be easily computed by writing the matrix of δ_k. In our particular case, we can write the matrix of δ_1, which base is given by the three vectors $[a,b], [b,c], [a,c]$, written in the basis $[a], [b], [c]$ of $C_0(K, \mathbb{Z}/2\mathbb{Z})$. Hence, we have the following matrix

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Therefore, since $rank(M) = 2 = \dim(\mathrm{Im}(\delta_1))$ and hence, $\dim(\mathrm{Ker}(M)) = 1 = \dim(\mathrm{Ker}(\delta_1))$, and we have that $\mathrm{Ker}(\delta_0) = C_0(K, \mathbb{Z}/2\mathbb{Z})$and $\mathrm{Im}(\delta_2) = 0$ we can determine H_0 and H_1 (obviously we have the same results as in the "hand-craft" computation above).

2.3 Equivalence between simplicial and singular homology

It is convenient for our work and particularly for the next section to point out that there is a significant theorem of algebraic topology which states that simplicial homology groups and singular homology groups are equivalent whenever both can be calculated. Since the latter is defined in every topological space, the main proof consists of showing an isomorphism between nth singular homology groups and nth simplicial homology groups. Notice that we do not consider those topological spaces that are not homeomorphic to a simplicial complex.

For further information and detail refer the reader to [p. 128][20].

2.4 Homotopy invariance

Another fundamental result provides a certain property of the pushforward of a continuous function in terms of homology. It is necessary to remind that a pushforward of a continuous function $f : X \to Y$ between topological spaces is a homomorphism $f_* : H_n(X) \to H_n(Y)$ for $n \geq 0$. This main result reveals that pushforwards are homotopy invariant, in the sense that if $f, g : X \to Y$ are homotopic, then $f_* = g_*$. This result immediately implies that homology groups of homotopic topological spaces are isomorphic.

For further detail and information refer the reader to [p. 110][20].

Chapter 3

Approximation of Topological Spaces from Data

Let $\mathcal{X}$ be a finite set of points lying in a metric space $(\mathbb{M}, d)$, i.e., a point cloud, and assume that $\mathcal{X}$ is sampled from a topological space $\mathbb{X}$. One of the core tasks in topological data analysis is to estimate the topology of the underlying space $\mathbb{X} \subset \mathbb{M}$ based on its sampled points $\mathcal{X}$ via homology. Once we have introduced the foundations of simplicial homology, it would be reasonable to define a topological scheme able to construct an abstract simplicial complex from $\mathcal{X}$ in a manner that the topological information of the underlying space $\mathbb{X}$ is kept.

Recall from section 2.3 and section 2.4 that homotopy equivalent topological spaces that are triangulable, or in other words, in which both singular and simplicial homology are correctly defined and coincide, can not be distinguished by homology. Hence, a reasonable goal would be to define the previously mentioned scheme so that its yielded simplicial complexes remain homotopy equivalent to $\mathbb{X}$ by construction. Notice that building simplicial complexes homotopy equivalent to another topological space homotopy equivalent to $\mathbb{X}$ would desirable too.

3.1 The Čech complex

In this subsection, we introduce the Čech complex, an abstract simplicial complex built from $\mathcal{X}$, capable of providing relevant information of the underlying topological space $\mathbb{X}$ under certain assumptions. To do so, first, we need to introduce the following concepts.

A covering $\mathcal{U} = \{U_i\}_{i \in I}$ of a finite set of points $\mathcal{X}$ lying in a metric space $(\mathbb{M}, d)$ is a family of subsets of $(\mathbb{M}, d)$ indexed by an index set I such that $\mathcal{X} \subseteq \bigcup_{i \in I} U_i$.

Definition 3.1. *The **nerve** of a covering $\mathcal{U} = \{U_i\}_{i \in I}$, denoted by $\mathcal{N}(\mathcal{U})$, is said to be* *the abstract simplicial complex (V, S) satisfying:*

- *$V = I$, i.e, the set of vertices of the abstract simplicial complex $\mathcal{N}(\mathcal{U})$ is the index set I indexing the covering $\mathcal{U}$.*
- *$\{i_0, ...i_n\}$ spans a n-simplex if and only if $\bigcap\limits_{i\in\{i_0,...i_n\}} U_i \neq \emptyset$.*

Generally, the nerve complex $\mathcal{N}(\mathcal{U})$ of some covering $\mathcal{U} = \{U_i\}_{i\in I}$ does not need to reflect any substantial information about the topology such covering. However, there exist certain conditions under which it reflects relevant information about the topology. The following theorem, called "the nerve theorem", provides such criteria that guarantees $\mathcal{N}(\mathcal{U})$ being homotopy equivalent to $\bigcup_{i\in I} U_i$.

Theorem 3.2 (Nerve Theorem). *Let $\mathcal{U} = \{U_i\}_{i\in I}$ be a covering of a finite set of points $\mathcal{X}$ lying in a metric space $(\mathbb{M}, d)$. Suppose that for every finite subset of $I, J \subset I$, the set $\bigcap\limits_{i\in J} U_i$ is either empty or contractible, then the geometric realization of $\mathcal{N}(\mathcal{U})$ is homotopy equivalent to $\bigcup_{i\in I} U_i$.*

Definition 3.3. *Let $\mathcal{X}$ be a point cloud in a metric space $(\mathbb{M}, d)$ satisfying $\mathcal{X} \subseteq \bigcup\limits_{x\in\mathcal{X}} B_\epsilon(x)$ for some $\epsilon > 0$, then $\check{C}_\epsilon(\mathcal{X}) = \mathcal{N}(\{B_\epsilon(x)\}_{x\in\mathcal{X}})$ is said to be the **Čech complex** attached to $\mathcal{X}$ and ϵ.*

Observe that the Čech complex of $\mathcal{X}$ does not need to be embedded in the same metric space in which $\mathcal{X}$ lies. To illustrate the latter, suppose $\mathcal{X} \subset \mathbb{R}^n$, and $|\mathcal{X}| = d$, with $d > n + 1$, then, the its associated Čech complex could have, for instance, a simplex of dimension $(d - 1)$, which obviously does not lie in $\mathbb{R}^n$.

Hence, in virtue of 3.2, in order to conclude that $\tilde{C}_\epsilon(\mathcal{X})$ is homotopy equivalent to the underlying space $\mathbb{X}$, it would be sufficient to prove that the union of restricted balls $(\bigcup\limits_{x\in\mathcal{X}} B_\epsilon(x))_\mathbb{X}$ covers $\mathbb{X}$ and that any arbitrary non-empty intersection of restricted balls is contractible. If $\mathcal{X}$ is a convex set, then the nerve theorem applies straightforwardly.

However, it is difficult to establish when the property mentioned above holds. It is not clear a priori what properties of $\mathcal{X}$ will imply it. That is why, for further information and detail, we refer the reader to a recent paper [24] in which they provide a more technical and complete exposition of certain conditions under which the latter is satisfied.

However, even though under certain conditions [24] $\tilde{C}_\epsilon(\mathcal{X})$ is homotopy equivalent to $\mathbb{X}$, their construction involves a highly computational cost.

The main computation regarding Čech complexes requires to compute the minimum enclosing ball. Let us refer the reader to [15] to check a robust, practical implementation.

Computing the minimum enclosing ball of K points can be linear in K, however, computing the Čech complex depends superexponentially on the dimension N of R^N. Let us refer the reader to [12] for further details of a Čech complex construction algorithm and complexity. Hence, it would be desirable to find a more computationally, relaxed scheme.

3.2 The Vietoris-Rips complex

In order to solve our main problem related to Čech complexes, we introduce the following concepts.

Definition 3.4. *A **flag complex** is an abstract simplicial complex that has no empty simplices; that is, there is no subset of vertices such that each pair of vertices belongs to a face of the complex, but the set of vertices is not a face of the complex itself.*

Remark 3.5. Observe that an abstract simplicial complex $K = (V, S)$ which satisfies the property of being a flag complex, only depends on its 1-skeleton, i.e., S_1.

The most used flag complex in our context is the **Vietoris-Rips complex**.

Definition 3.6. *Let $\mathcal{X}$ be a point cloud in some metric space. Then the **Vietoris-Rips complex** attached to $\mathcal{X}$ and δ for a certain $\delta > 0$, denoted by $VR_\delta(\mathcal{X})$, is the abstract simplicial complex with vertex set $\mathcal{X}$ such that spans a n-simplex for each subset of $\mathcal{X}$ with cardinality $n + 1$ and diameter δ.*

Hence, observe that the Vietoris-Rips complex reconstruction involves less computational cost than the Čech complex. As we have stated above, the 1-skeleton of the Vietoris-Rips determines its whole structure. Let us refer the reader to [46] to check an efficient implementation of the Vietoris-Rips reconstruction. Observe that the Vietoris-Rips complex can be built in any metric space.

Moreover, in the same way, we did for Čech complexes, let us refer the reader to [24] for further details about the homotopy reconstruction via Vietoris-Ribs complex and the conditions under which it is homotopy equivalent to the target space $\mathbb{X}$.

3.3 Interleaving relation

At this point, we should have noted that Vietoris-Rips complexes do not generally inherit the same properties for topological reconstruction than Čech complexes do.

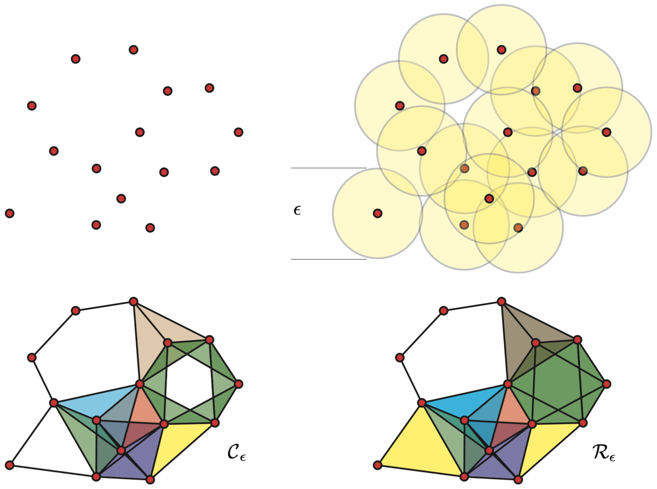


Figure 3.1: Čech complex reconstruction (lower -left) and Vietoris Rips reconstruction (lower-right) over a set of points for a certain ϵ parameter. Extracted from [16].

However, the Vietoris-Rips can approximate the Čech complex. In fact, the Vietoris-Rips complex and the Čech complex have the following interleaving relationships.

Lemma 3.7. *For a point cloud $\mathcal{X}$ in some metric space, $\tilde{C}_\epsilon(\mathcal{X}) \subseteq VR_{2\epsilon}(\mathcal{X}) \subseteq \tilde{C}_{2\epsilon}(\mathcal{X})$.*

Let us refer the reader to [40] for the proof of 3.7.

In fact, the following is fulfilled.

Lemma 3.8. *For a point cloud $\mathcal{X} \in \mathbb{R}^n$, $\tilde{C}_\epsilon(\mathcal{X}) \subseteq VR_{2\epsilon}(\mathcal{X}) \subseteq \tilde{C}_{\sqrt{\frac{2n}{n+1}}\epsilon}(\mathcal{X})$.*

Let us refer the reader to [40] for the proof of 3.8.

Chapter 4

Persistence Homology

As stated in previous sections, homology groups are topological invariants, i.e., if two topological spaces are homeomorphic, they have isomorphic homology groups. Hence, they are sensible to continuous transformations of the topological space under consideration. Moreover, in the previous chapter, we presented the Vietoris Rips complex, the topological scheme that we will use to build abstract simplicial complexes structures from data which intrinsically depends on a parameter δ.

At this point, one may notice that we will need a tool capable of dealing with two basic problems while reconstructing the topological features of a space from a sample of it. First, we have noisy data which could provide an inaccurate representation of the real space under study. Second, small alterations of the parameter δ can definitely modify the resulting simplicial complex.

Persistence homology first appears in [14], to formalize and define a topological invariant for sequences of simplicial complexes linked via simplicial maps which could be capable of summing up the induced homological information. Intuitively, it provides a topological invariant encompassing the homology groups induced by all these possible perturbations of δ. Then, the main idea is motivated by the fact that significant topological features persist on a wide range of parameters.

This chapter is mainly based on the publications [47], and [8] and will provide the theoretical foundations of persistence homology.

Recall that R the integral domain $\mathbb{Z}$, or an arbitrary field $\mathbb{F}$.

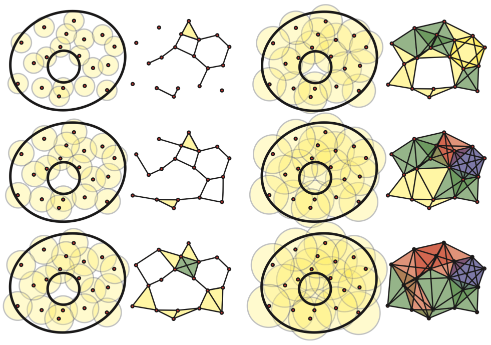


Figure 4.1: Vietoris-Rips complex reconstruction encoding different for diferent ϵ parameters. Observe that different topologies appear. Extracted from [16].

4.1 From Vietoris-Rips complex to $\mathbb{N}$-persistence modules

4.1.1 Filtration

First of all, it is convenient to define the notion of filtration since it is widely used in persistence articles. In fact, some further stability results of certain topological summaries that will be latter discussed are usually given by employing functions f such as the one described in the following definition.

Definition 4.1. *Let K be a simplicial complex, and $f : K \to \mathbb{R}$ a monotonic function, i.e, $\sigma \subseteq \tau$ for $\sigma, \tau \in K$ implies $f(\sigma) \leq f(\tau)$. Then, due to the monotonicity of the function f, we are able to construct subcomplexes of K for all $a \in \mathbb{R}$ as follows $K(a) = f^{-1}(-\infty, a]$. Hence, having $a_1 < a2 < ... < a_n$ function values and $a_0 = -\infty$, by denoting $K(a_i) = K_i$, we can form the following increasing sequence called **filtration**:*

$$\emptyset = K_0 \subseteq K_1 \subseteq ... \subseteq K_n = K.$$

However, in our next discussion, we will employ another kind of construction to understand and define persistence homology.

4.1.2 Vietoris-Rips complex and $\mathbb{R}$-persistence objects

Let us refer the reader to [26] to understand the following category theory notation. Given a partially ordered set P, there is a corresponding category $\mathcal{P}$ whose objects are the elements of the set P; that is, obj($\mathcal{P}$) = P, and whose morphisms link an element A to an element B whenever $A \leq B$. Then, let $\mathcal{C}$ be a category and P a partially order set with its corresponding category $\mathcal{P}$ constructed as above. A $\mathcal{P}$-persistence object in C is a functor $\Pi : \mathcal{P} \to \mathcal{C}$. Therefore, it is a family

$\{C_X\}_{X\in obj(\mathcal{P})}$ of objects of $\mathcal{C}$ together with a family of morphisms $\pi_{XY} : C_X \to C_Y$ whenever $X \leq Y$. $\mathcal{P}$-persistence objects form a category by their own, where every morphism correspond with a natural transformation. These category is denoted by $\mathcal{P}_{pers}(\mathcal{C})$.

Recall that if $\delta \leq \delta'$ then $VR_\delta(\mathcal{X}) \subseteq VR_{\delta'}(\mathcal{X})$; the same states for Čech complexes. Hence, note that of Vietoris-Rips or Cech complexes yield an $\mathbb{R}$**-persistence simplicial complex** if equipped with inclusion maps $VR_\delta(\mathcal{X}) \hookrightarrow VR_{\delta'}(\mathcal{X})$ whenever $\delta \leq \delta'$. Moreover, the associated sequence of chain complexes over R connected by inclusion maps form an $\mathbb{R}$**-persistence chain complex** over R. Hence, the homology of an $\mathbb{R}$-persistence chain complex is an $\mathbb{R}$**-persistence module** over R.

Recall that in section 2.2.2 we stated the homology groups decomposition over R. Those structure theorems make homology very useful and easy to compute. Thus, in the same way, it would be highly desirable to have an easy decomposition structure for $\mathbb{R}$-persistence modules over R. However, there is just a classification theorem for a subcategory of the category of $\mathbb{N}$-persistence $\mathbb{F}$-vector spaces, more concretely, $\mathbb{N}$-persistence $\mathbb{F}$-vector spaces of finitely generated type. Therefore, first of all, we need to find a way to redefine Vietoris-Rips complex as $\mathbb{N}$-persistence simplicial complexes, i.e., filtrations.

4.1.3 From Vietoris-Rips complexes to $\mathbb{N}$-persistence objects

Since we work with finite point clouds, the distance function on our metric space takes just finitely many values, and thus, there are finitely many real values in which the structure of our simplicial complex changes. Therefore, we can enumerate those distances $\{\delta_0, ..., \delta_n\}$ and define an order preserving map $p : \mathbb{N} \to \mathbb{R}$ such that $g(i) = \delta_i$ if $i \leq n$ and $g(i) = \delta_n$ if $i > n$. Hence, we can consider the family of Vietoris-Rips complexes $\{VR_{\delta_i}(\mathcal{X})\}_{i=0\div n}$ attached to a point cloud $\mathcal{X}$. The collection $\{VR_{\delta_i}(\mathcal{X})\}_{i=0\div n}$ together with inclusion maps $\pi_{i,i+1} : VR_{\delta_i}(\mathcal{X}) \hookrightarrow VR_{\delta_{i+1}}(\mathcal{X})$ form a $\mathbb{N}$-persistence simplicial complex, or in other words, a filtration of K. Hence, we hace the following situation:

$$VR_{\delta_0}(\mathcal{X}) \xrightarrow{\pi_{0,1}} VR_{\delta_1}(\mathcal{X}) \xrightarrow{\pi_{1,2}} ... \xrightarrow{\pi_{n-1,n}} VR_{\delta_n}(\mathcal{X}).$$

Note that for each VR_{δ_i} we have associated a chain complex $(C^i_*(K,R), \delta_*)$, and thus, our $\mathbb{N}$-persistence simplicial complex induces a $\mathbb{N}$-persistence chain complex. Moreover, such inclusion maps induce group homomorphisms on homology, $H_n(VR_{\delta_i}(\mathcal{X}), R) \to H_n(VR_{\delta_{i+1}}(\mathcal{X}), R)$ and hence, a $\mathbb{N}$-persistence module.

Definition 4.2. *An* $\mathbb{N}$*-persistence module is said to be of* ***finitely generated type*** *if every module* M_i *is finitely generated and there exists an* $m \in \mathbb{N}$ *such that for all* $i \geq m$*,*

$\pi_{i,i+1} : M_i \to M_{i+1}$ is an isomorphism.

Observe that the above-mentioned $\mathbb{N}$-persistence modules induced by the $\mathbb{N}$-persistence Vietoris-Rips complex are of finite type by construction.

4.2 Correspondence of $\mathbb{N}$-persistent modules of finite type over R and R[t]-modules

Before starting the correspondence theorem, which is the main result of this subsection, we need to introduce the following concepts.

Definition 4.3. *A **graded ring** is a ring $(R, +, \cdot)$ equipped with a direct sum decomposition of abelian groups $R \cong \bigoplus_i R_i$, $i \in \mathbb{Z}$, so that $R_n R_m \hookrightarrow R_{n+m}$.*

Definition 4.4. *A **graded module** over a graded ring R is a module equipped with a direct sum decomposition $M \cong \bigoplus_i M_i$, $i \in \mathbb{Z}$, so that $R_n M_m \hookrightarrow M_{n+m}$.*

At this point, after having defined the previous concepts of the subsection, we are allowed to define do the following construction or assignment. Let denote the $\mathbb{N}$-persistence module as $\mathcal{M} = \{M^i, \pi_{i,i+1}\}_{i\geq 0}$ and let $R[t]$ be equipped with the standard grading, i.e, $R_n = Rt^n$ for $n \geq 0$. We then define a **graded module over $\mathbf{R}[\mathbf{t}]$** as

$$\alpha(\mathcal{M}) = \bigoplus_{i=0}^{\infty} M^i,$$

where the R-module structure is simply the sum of the structures on the individual components and where the action of t is given by

$$t(m_0, m_1, ...) = (0, \pi_{0,1}(m_0), \pi_{1,2}(m_1), ...).$$

In other words, t shifts elements of the module up in gradation.

Theorem 4.5 (Correspondence). *Let R be a commutative ring with unity. The category of $\mathbb{N}$-persistence modules of finitely generated type over R is equivalent to the category of finitely generated graded modules over $R[t]$.*

Given the assignment α above, there is a proof of the equivalence stated in Theorem 4.5 of [48] based on the Artin-Rees theory in commutative algebra (Eisenbud, 1995). For further explanation and a generalization of the Theorem 4.5 let us refer the reader to [34].

4.3 Decomposition

The importance of the theorem above arises when R is a field $\mathbb{F}$. In such case, graded $R[t]$-modules (finitely generated persistence modules) allow the decomposition stated in the following theorem:

Theorem 4.6 (Structure Theorem). *Let D be a principal ideal domain. Then, every finitely generated module M over D is isomorphic to a direct sum of cyclic D-modules. That is, there is a unique decreasing sequence of proper ideals $(d_1) \supseteq (d_2) \supseteq \cdots \supseteq (d_m)$ such that it decomposes into the form:*

$$M \cong D^{\beta} \oplus \left(\bigoplus_{i=1}^{m} D/(d_i) \right), \tag{4.1}$$

where $d_i \in R$, and $\beta \in \mathbb{Z}$.

Moreover, every graded module over a graded principal ideal domain D, decomposes uniquely into the form:

$$M \cong \left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} D \right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} D/(d_j) \right), \tag{4.2}$$

where $d_j \in R$ are homogenous elements such that $(d_1) \supseteq (d_2) \supseteq \cdots \supseteq (d_m), \alpha_i, \gamma_j \in \mathbb{Z}$, and Σ^{α} denotes an α-shift upward in grading.

Remark 4.7. A classic algebra result states that if D is a domain, then $D[x]$ is a principal ideal domain, i.e., a PID, if and only if, D is a field.

From 4.7, observe that $\mathbb{Z}[x]$ is not a PID since $\mathbb{Z}$ is not a field. Thus, as Theorem 4.5 suggests, there is no simple classification for persistence modules over $\mathbb{Z}$. Hence, as mentioned at the end of the subsection 2.2, in persistent homology, it is convenient to work with coefficients over a field. Note that in the case of graded $\mathbb{F}[t]$-modules, ideals are of the form $(t)^n = t^n$, and hence, (4.2) can be rewritten as

$$\left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} F[t] \right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} F[t]/(t^{n_j}) \right). \tag{4.3}$$

Definition 4.8. *We will say that a **$\mathcal{P}$-interval** is a pair (i,j) with $0 \leq i < j \in \mathbb{Z} \cup \infty$.*

Hence, using the decomposition result (4.2) of Theorem 4.6 for a graded $\mathbb{F}$[t]-module, we can define a bijection U from the latter to a set $\mathcal{S}$ of $\mathcal{P}$-intervals in the following way:

Let $\mathcal{S}$ = $\{(i_1, j_1), (i_2, j_2), ..., (i_n, j_n)\}$ be the set of $\mathcal{P}$-intervals. Then, let us define

$$U(i,j) = \begin{cases} \Sigma^i F[t] \text{ if } j = \infty \\ \\ \Sigma^i F[t]/(t^{j-i})\text{otherwise} \end{cases}$$

Finally, define

$$U(\mathcal{S}) = \bigoplus_{l=1}^{n} U(i_l, j_l).$$

Observe that at this point we have defined a bijection between the finite sets of $\mathcal{P}$-intervals and the finitely generated graded modules over the graded ring $\mathbb{F}[t]$. Recall that Theorem 4.5 stated in the previous section establishes a correspondence between the category of persistence modules of finitely generated type and the category of finitely generated graded modules over $R[t]$. Consequently, the isomorphism classes the isomorphic classes of $\mathbb{N}$-persistence modules over $\mathbb{F}$ of finite type are in bijective correspondence with the finite sets of $\mathcal{P}$-intervals.

Observe that each $\mathcal{P}$-interval describes an element of the basis of all homology groups $H_k^s(VR_{\delta_k}(\mathcal{X})), i \leq s \leq (j-1)$. In other words, each element is a k-cycle e that is completed at the time i, forming a new homology class that remains non-bounding until time j, at which time it joins the boundary group B_k. Therefore, usually, the elements of the finite sets of $\mathcal{P}$-intervals, (i_l, j_l) , are denoted as (b_i, d_i) being b_i the **birth of the i-th homology feature** and d_i the **death of the ith homology feature**, and its persistence.

4.4 Barcodes and Persistent Diagrams

4.4.1 Barcodes

The latter may inspire a visual snapshot in the form of a **barcode**. That is a graphical representation in the plane of $H_*(VR_{\delta_*}(\mathcal{X}), \mathbb{F}))$ as horizontal segment lines, where the horizontal axis represents the value of the parameter δ and the vertical axis an arbitrary order of homology generators. Moreover, note that the free parts of the decomposition stated in the above section correspond to the barcode's infinite bars, and the torsional parts of the decomposition correspond to the finite bars of the barcode.

4.4.2 Persistence Diagrams

Another convenient representation, is the **persistence diagram**. Persistence diagrams are topological summaries in bijective correspondence with barcodes,

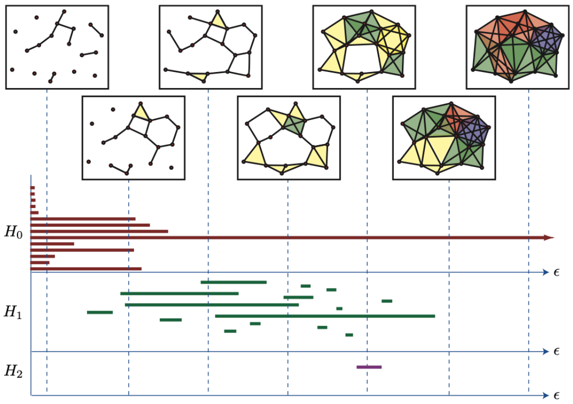


Figure 4.2: Barcode visual snapshot of the H_0, H_1, H_2 homology groups. Extracted from [16].

defined as $\mathcal{D} = (\mathcal{S} = \{(b_1, d_1), ..., (b_n, d_n)\} \subset \mathbb{R}^2)$, i.e, finite sets of $\mathcal{P}$-intervals embedded in $\mathbb{R}^2$.

At least in part, the interest in persistent homology intends to generate data summaries revealing relevant information that, otherwise, classical data analysis methods can not provide. As we have seen, different features, and in particular, homology generators are tracked resulting in an object known as persistence diagram. Therefore, since we want to create an alternative framework for data analysis, it is fair to ask ourselves questions such as the following. Given a collection of point cloud realizations of the same topological space, which is the average of its corresponding persistence diagrams? Do small variations in the point cloud result in little changes in its associated persistence diagram, or other words, are they stable?

Now we will intend to briefly explain some of the work that has been done to answer these questions. In order to measure similarities between the following definition of distance ha been used.

Definition 4.9. *Let be D_k^1, D_k^2 persistence diagrams. The pth Wasserstein distance between them two (Bottleneck distance if $p = \infty$) is defined as*

$$W_p(D_k^1, D_k^2) = \inf_{\gamma : D_k^1 \to D_k^2} \left[\sum_{a \in D_k^1} \| a - \gamma(a) \|_\infty^p \right]^{1/p}$$

where the infimum is over all possible bijections $\gamma : D_k^1 \to D_k^2$.

The set of persistence diagrams $\mathcal{D}$ endowed with the p-Wasserstein distance

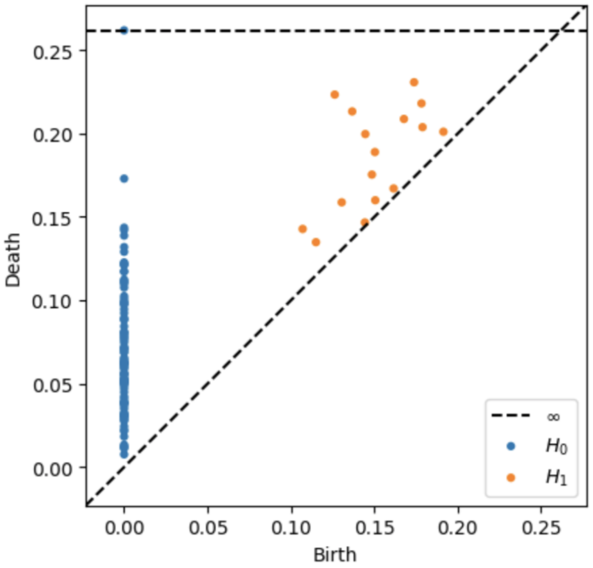


Figure 4.3: Random data H_0 and H_1 persistence diagram plot using Ripser.py

form a metric space $(\mathcal{D}, W_p)$. However, the latter turns out to be incomplete [27], and therefore, not very appropriate for inference.

Recall that a metric space is said to be complete if for all d-Cauchy sequence $(x_n)_{n=1}^{\infty}$ in (X, d) it exists an element $x \in X$ such that $\lim_{n\to\infty} \| x_n - x \| = 0$.

Example 4.10. Let $x_n = (0, 2^n) \in \mathbb{R}^2, n \in \mathbb{N}$ and let D_n be the persistence diagram containing $x_1, ..., x_n$ (each with multiplicity 1). Then $W_p\left(D_n^1, D_{n+k}^2\right) \leq \frac{1}{2^{n+k}}$, so D_n is Cauchy. However, the number of off-diagonal points in D_n grows to ∞ as $n \to \infty$, so the limit of the sequence is not in $(\mathcal{D}, W_p)$. Hence, $(\mathcal{D}, W_p)$ is incomplete.

To solve the incompleteness, the **generalized persistence diagram** is defined in [27] as $\mathcal{D} = (\mathcal{S} \subset \mathbb{R}^2) \cup \Delta$, for $\Delta = \{(x,x) \in \mathbb{R}^2 | x \in \mathbb{R}\}$, and the space of persistence diagrams as $\mathcal{D}_p = \{D : W_p(D, D_\phi) < \infty\}$, being $D_\phi = \Delta$, the empty persistence diagram.

Under the above construction, stability can be stated and proven. Before referring the reader to [9] we introduce the following concepts.

Definition 4.11. *Given two metric spaces (X, d_X) and (Y, d_Y), a function $f : X \to Y$ is said to be **L-Lipschitz** or **Lipschitz continuous** if there exists a constant $L \in \mathbb{R}$, $L \geq 0$, such that for all x_1, $x_2 \in X$, $d_Y(f(x_1), f(x_2)) \leq Ld_X(x1, x2)$. In this case, L is referred as a **Lipschitz constant**. If L is the smallest constant satisfying the latter, it is said to be the best Lipschitz constant $Lip(f)$. Moreover, if $L = 1$, f is said to be a short map, and if $L \in [0, 1)$, f is said to be a contraction.*

A crucial property of Lipschitz functions is that their level sets are well separated. In fact, the distance between the level sets defined by values $a \leq b$ is at least the difference in values divided by the Lipschitz constant, $d(x,y) \geq \frac{(b-a)}{Lip(f)}$ whenever $f(x) = a$ and $f(y) = b$, or in other words that corresponding simplicial complexes differ in points at distance at most $\frac{(b-a)}{Lip(f)}$.

At this point, let's recall the construction on 4.1.1, and observe that nested family of sublevel sets defines a sequence of homology groups analogous to the ones defined in the previous subsection. We say that f is **tame** if this sequence is finite and consists of homology groups whose ranks are finite. The latter is analogous to ask persistence modules be of finitely generated type.

We can now refer the reader to [9] to check the Wasserstein stability theorem, that shows that under mild assumptions on the topological space $\mathbb{X}$, computing a persistence diagram of a tame Lipschitz function defined as above is a continuous map. Moreover, let refer the reader to [13] for further discussion on persistence stability theorems.

On the other hand, much effort has been made to build a proper environment where persistence diagrams could work well in terms of expectations, variances and covariances. However, we have not obtained the desired results. For instance, in [27] it is proven that $\mathcal{D}_p$ is, in fact, a Polish space, i.e., a space homeomorphic to a complete metric space that has a countable dense subset, i.e., separable ($\{x_n\}_{n=1}^{\infty}$ such that every nonempty subset of the space contains at least one element of the sequence). We will not deem in why Polish spaces provide the correct framework for the latter, we will say with no rigour, that measurable functions between Polish spaces always take Borel sets to analytical sets which are universally measurable, and hence, everything "works fine". After having shown the latter, Fréchet expectation and variance definition for persistence diagrams over the probability space $(\mathcal{D}_p, \mathcal{B}(\mathcal{D}_p), \mathcal{P})$ for a given probability measure [Definition 22][27] is provided. However, the latter definition involves a minimization, and hence, expectation existence and uniqueness are not guaranteed. For the former, certain regularity conditions on $\mathcal{P}$ under which existence is ensured are proposed in [27]. Additionally, in [28], a solution that uses a probabilistic approach is proposed to define a unique Fréchet average, but its computation remains practically prohibitive. Therefore, persistence diagrams apparently do not work properly for statistical analysis.

Before stepping into the next section it is necessary to remark that in [27] it is shown how the probability space $(\mathbb{X}, \mathcal{B}(\mathbb{X}), \mathcal{P}_\theta)$ with $\mathbb{X} \subset \mathbb{R}^d$ and $\mathcal{P}_\theta$ a probability measure from which point cloud data is randomly generated ($X(\omega)$is a point cloud) induces a measure P_D on $(\mathcal{D}_p, \mathcal{B}(\mathcal{D}_p))$.

Chapter 5

Introduction to Stochastic Processes

This chapter aims to make a brief introduction to Stochastic Processes and its basic concepts. The vast majority of the concepts have been extracted from [19] and [45], the latter being provided by its author and thesis advisor. We need to make this parenthesis because first, stochastic processes will appear in the form of convergence results in the following chapter, and even more important, they will serve us as a pillar for our last chapter.

A time series is a series of observations x_t observed over a period of time. In our work, we will focus on the case that observations are made at fixed equidistant time points $\{x_t : t \in \mathbb{Z}\}$.

The progress in time series analysis requires to introduce the following probabilistic model definition:

Definition 5.1. *Given a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, where Ω is a sample space, $\mathcal{F}$ a σ-algebra and $\mathcal{P}$ a probability measure, and a measurable space (S, Σ), a* ***stochastic process*** *is a collection of S-valued random variables which can be written as $\{X(t, \omega) : t \in T, \omega \in \Omega\}$.*

Definition 5.2. *Given a stochastic process $\{X(t, \omega) : t \in T, \omega \in \Omega\}$, the mapping $X_w : T \to S$ is called a* ***realization*** *of the stochastic process.*

In other words, a realization is a single outcome of the stochastic process. In our work, we will interpret T as time, so a realization of the stochastic process will represent a time series. In fact, we are going to take into account just discrete-time stochastic processes, in which $T = \mathbb{N}$ or $\mathbb{Z}$. In such case we can write $\{X_t : t \in T\}$.

Definition 5.3. *Given a stochastic process $\{X_t : t \in \mathbb{N}\}$, we say it is a* ***second order stochastic process*** *if it satisfies*

$$\mathbb{E}[X_t^2] < \infty, \ \forall t \in \mathbb{N}$$

Remark 5.4. The definition 5.3 guarantees that $\mathbb{E}[X_t]$ and $\mathbb{V}[X_t]$ are finite and well defined.

Applying Jensen's inequality

$$|\mathbb{E}[X_t]| \leq \mathbb{E}[|X_t|] < \infty$$

together with Cauchy-Schwarz inequality

$$\mathbb{E}[|X_t|] \leq \mathbb{E}([X_t^2])^{\frac{1}{2}} < \infty,$$

we get that

$$\mathbb{V}[X_t] := \mathbb{E}[X_t^2] - (\mathbb{E}[X_t])^2 < \infty.$$

It guarantees that for any two random variables of the process, the covariance is well defined. Applying the Cauchy-Schwarz inequality, we have

$$\begin{aligned} |\mathbb{C}[X_t, X_{t+l}]| &= |\mathbb{E}[(X_t - \mathbb{E}[X_t])(X_{t+l} - \mathbb{E}[X_{t+l}])]| \\ &\leq \mathbb{E}[|(X_t - \mathbb{E}[X_t])||(X_{t+l} - \mathbb{E}[X_{t+l}])|] \\ &\leq \mathbb{E}[|X_t - \mathbb{E}[X_t]|^2]^{\frac{1}{2}}\mathbb{E}[|X_{t+l} - \mathbb{E}[X_{t+l}]|^2]^{\frac{1}{2}} \\ &= \mathbb{V}[X_t]^{\frac{1}{2}}\mathbb{V}[X_{t+l}]^{\frac{1}{2}} \qquad (5.1) \end{aligned}$$

From now on, we assume that all stochastic processes appearing in this work are second-order stochastic processes.

5.1 Stationarity

To be useful, in most cases, stochastic processes must preserve certain probabilistic properties during time.

Definition 5.5. *A stochastic process $\{X_t : t \in \mathbb{Z}\}$ is said to be* ***(weakly) stationary*** *it satisfies the following properties:*

1. $\mathbb{E}[X_t] = \mu$, $\forall t \in \mathbb{Z}$

2. $\mathbb{C}[X_t, X_{t+l}] = \gamma(l)$, $\forall t, l \in \mathbb{Z}$. *By the symmetry of the covariance, $\gamma(-l) = \gamma(l)$, thus, $\gamma(l)$ can be defined in $\mathbb{N}$. We call $\gamma(l)$ the* ***autocovariance function*** *(ACVF) and the value "l", the "lag".*

Remark 5.6. Note that in the case $l = 0$, $\gamma(0) = \mathbb{C}[X_t, X_{t+0}] = \mathbb{C}[X_t, X_t] = \mathbb{V}[X_t]$.

Definition 5.7. *Given a stationary stochastic process $\{X_t : t \in \mathbb{Z}\}$, the auto-correlation function is said to be*

$$\rho(l) := \frac{\gamma(l)}{\gamma(0)}, \forall k \in \mathbb{Z}.$$

Notice that since $\gamma(l)$ can be defined in $\mathbb{N}$, so can be $\rho(l)$.

Remark 5.8. From 5.1, we have that $|\gamma(l)| \leq |\gamma(0)|, \forall l \geq 0$. Hence, $-1 \leq \rho(l) \leq 1$.

Definition 5.9. *Given a stochastic process $\{X_t : t \in \mathbb{Z}\}$, we say that it is **strictly stationary** if for any $\{t_1, ..., t_n\} \in \mathbb{Z}$, and l, the vectors $(X_{t_1}, ..., X_{t_1})$ and $(X_{t_{1+l}}, ..., X_{t_{n+l}})$, have the same law.*

Note that the above implies that all the random variables X_t have the same law too. Hence, it is obvious that if a stochastic process is strictly stationary, it is stationary too. The reverse is false in general.

Before ending this chapter, let us give to common and basic examples of stochastic processes.

Example 5.10. (**IID Noise**) We say that $\{X_k, k \geq 1\}$ is an i.i.d noise if all the random variables are independent with mean μ and standard deviation σ. In this case $\rho(l) = 0$ if $l \geq 1$, i.e, is a completely random time series.

Example 5.11. (**White Noise**) We say that $\{X_k, k \geq 1\}$ is white noise if all the random variables have expectation μ and standard deviation σ and $\rho(l) = 0$ if $l \geq 1$. Observe that IID is a particular case of White Noise.

Example 5.12. (**Gaussian process**) We say that a process $\{X_t, t \in T\}$ is Gaussian if and only if, for every set of indices $t_1, ..., t_k$ in the index set T, $(X_{t_1}, ..., X_{t_k})$ is a multivariate Gaussian random variable, or equivalently, every lineal combination of $(X_{t_1}, ..., X_{t_k})$ has a univariate normal distribution.

Chapter 6

Persistence Diagrams Functional Summaries

As we have already mentioned, one of the central workflows in TDA is to start with data point cloud and compute a topological summary of this to provide useful information about its structure and geometry. Therefore, as stated in [27], one of the worst dangers would be to ignore the fundamental aspect of classical data analysis such as expectations, variances and conditional probabilities. In 4.4.2, we ended by asserting that persistence diagrams sometimes tend to be challenging for achieving certain statistical goals; in fact, they do not have a unique mean. In order to deal with the latter many functional summaries have appeared. Then, the persistence diagram's analysis becomes a statistical function analysis.

6.1 Introduction to functional summaries

At this point, suppose that we have $\mathcal{D}_1, ...\mathcal{D}_n$ persistence diagrams generated from the same population. From [27], we can suppose a probability distribution $\mathcal{P}$ on the probability space $(\mathcal{D}_p, \mathcal{B}(\mathcal{D}_p), \mathcal{P})$ and suppose $\mathcal{D}_1, ...\mathcal{D}_n$ as random variables independently and identically distributed on the latter space. Let $\mathcal{F}$ be a collection of functions. Then, a **functional summary** $\mathbb{F}(D)$ is a map

$$\mathbb{F} : \mathcal{D} \to \mathcal{F}.$$

Then, $F_i = \mathbb{F}(\mathcal{D}_i)$ can be understood as a random variable too. In fact, since $\{\mathcal{D}_i\}_{1 \leq i \leq n}$ are independent and identically distributed random variables,

$$F_1, ...F_n \sim^{i.i.d} \mathcal{P}_{\mathcal{F}}.$$

We could think of obtaining a sample mean functional summary to serve us as an estimator of the population mean. In fact, we can define the mean functional summary as

$$\bar{F}(t) = \mathbb{E}_{\mathcal{P}_{\mathcal{F}}}[F_i(t)].$$

Moreover, we can define the pointwise estimator as

$$\hat{F}(t) = \frac{1}{n}\sum_{i=1}^{n} F_i(t).$$

Let denote $\mathcal{B}_F$ by the set of all possible functions given a functional summary and let $\mathbb{T}$ be the compact where we want to compute $\bar{F}(t)$. Suppose that $F(t) = 0$ for all $t \notin \mathbb{T}$ for all $F \in \mathcal{B}_F$. Moreover, assume that there exists $\bar{U} < \infty$ and that

$$\sup_{\mathcal{F}\in\mathcal{B}_{\mathcal{F}}} \sup_{t\in\mathbb{T}} \mid F(t) \mid \leq \bar{U}. \tag{6.1}$$

With the above construction, the following two propositions are stated and proven in [3].

Proposition 6.1 (Pointwise Convergence). *Assume 6.1 holds. If $\mathcal{B}_F$ is an equicontinious set, then*

$$\sup_{t\in\mathbb{T}} \mid \hat{F}(t) - \bar{F}(t) \mid \underset{a.s}{\longrightarrow} 0. \tag{6.2}$$

In fact, if there exists L constant such that any $F \in B_F$ is L-Lipschitz, 6.2 holds too.

We need to introduce the following notations to state the next proposition; however, we will not delve into them. The main result for our discussion, in both propositions, is regarding the implication of functional summaries being L-Lipschitz.

Let $\mathcal{W} = \{\mathbb{F}_t : t \in \mathbb{T}\}$ such that $\mathbb{F}_t(D_i) = F_i(t)$ for $1 \leq i \leq n$. Let $\mathcal{Q}$ be a probability measure over $\mathcal{B}_F$ and $\parallel f - g \parallel_{\mathcal{Q},2} = \sqrt{\int \mid f(t) - g(t) \mid^2 d\mathcal{Q}(t)}$ be the $L_2(\mathcal{Q})$ norm for functions and let $\mathcal{N}(\mathcal{W}, L_2(\mathcal{Q}), \epsilon)$ the ϵ-covering number of $\mathcal{W}$.

Proposition 6.2. *Let $\sigma^2(t) = Var(F_i(t))$ and $\sigma^2 = \int \sigma(t)dt$. Assume 6.1 holds, then*

$$\sqrt{n}(\hat{F}(t) - \bar{F}(t)) \xrightarrow{D} N(0, \sigma^2(t))$$
$$\sqrt{n}(\int (\hat{F}(t) - \bar{F}(t))dt \xrightarrow{D} N(0, \sigma^2).$$

Moreover, if

$$\int_0^1 \sqrt{\log \sup_{\mathcal{Q}} \mathcal{N}(\mathcal{W}, L_2(\mathcal{Q}), \epsilon)} d\epsilon < \infty, \tag{6.3}$$

where $\bar{U}$ is the upper bound of the functional summary and the supremum is taken over all finitely discrete probability measures on the space of persistence diagrams, then $\sqrt{n}(\hat{F} - \bar{F})$ converges in distribution to $\mathbb{B}$, where $\mathbb{B}(t)$ is a Gaussian process over $t \in \mathbb{T}$ with a covariance function

$$\mathbb{C}[\mathbb{B}(t), \mathbb{B}(s)] = \mathbb{E}[F_i(t)F_i(s)] - \bar{F}(t)\bar{F}(s),$$

for $t, s \in \mathbb{T}$. Furthermore, if there exists a constant $L > 0$ such that any $F \in \mathcal{B}_F$ is L-Lipschitz, then the above three convergences hold.

In other words, if the functional summary is L-Lipschitz, then 6.3 holds, and therefore $\sqrt{n}(\hat{F} - \bar{F})$ converges to a Gaussian process.

Observe that at this point, we have presented a framework for generating convenient functional representations of persistence diagrams for statistical inference.

6.2 Persistence Landscapes and Silhouettes

In this section, first, we will define one of the most extended and used functional summaries, persistence landscapes, which will be an essential tool for developing our work. Persistence landscapes were introduced in [5] and followed by [6] as a well-behaviour summary for persistence diagrams. More precisely, they are functional summaries mapping persistence diagrams into elements of usually Banach, or Hilbert spaces.

Recall that a Banach space is a complete normed space, i.e., a pair $(X, \| \cdot \|)$ where X is a vector space over a field K, and $\| \cdot \|$ is a norm inducing a distance $d(x,y) = \| y - x \|$ under which X equipped with d is a complete metric space. Moreover, recall that an space H is called a Hibert space if it is an inner product space such that it is a complete metric space with respect to the distance function induced by the inner product.

In order to define persistence landscapes, we need to introduce the following triangle function.

Given an $\mathbb{N}$-persistence module of finitely generated type over a field $\mathbb{F}$, and its corresponding persistent diagram $\mathcal{D} = \{p_i(b_i, d_i)_{i \in I}\}$, let us define the following associated function:

$$\Lambda_p(t) = \begin{cases} t - b & t \in [b, \frac{b+d}{2}] \\ d - t & t \in [\frac{b+d}{2}, d] \\ 0 & otherwise \end{cases}$$

We define a **persistence landscape** as the following sequence of functions:

$$
\begin{aligned}
\mathbb{F}_k(D;t) = \ \eta(k,t) : \mathbb{N}\times\mathbb{R} : &\longrightarrow [0,+\infty] \\
(k,t) &\longmapsto kmax\{\Lambda_{p_i}(t), i = 1,...,m\}
\end{aligned}
$$

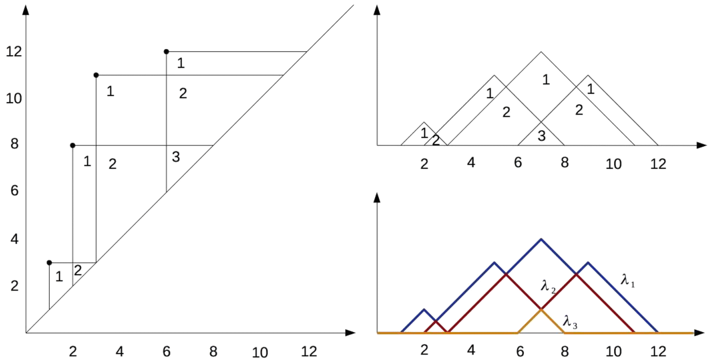


Figure 6.1: Peristence landscapes construction from a persistence diagram

Remark 6.3. These properties follow directly from the definition:

1. If $k > m$, then $\eta(k,t) := 0$.

2. $\eta(k,t) \geq 0$, for all k,t.

3. $\eta(k,t) \geq \eta(k+1,t)$, for all k,t.

Remark 6.4. Observe that $\Lambda_p(t)$ are one-Lipschitz functions, since$|\Lambda_p(t_2) - \Lambda_p(t_1)| \leq |t_2 - t_1|$ for all $t_2, t_1 \in \mathbb{R}$. Hence, persistence landscpaes $\eta(k,\cdot)$ are, by construction, one-Lipschitz too.

Persistence landscapes satisfy some very interesting properties such as invertibility [4]. Moreover, they are stable too, in the sense that, given two persitent diagrams, D_1, D_2 with their corresponding persistent landscapes η_1 and η_2, $|\eta(k,t)_1 - \eta(k,t)_2| \leq d_B(D_1, D_2)$, being d_B the bottleneck distance. Furthermore, they are parameter free, nonlinear and have good computational behaviour. Let us refer the reader to [7] for an implementation of it. Moreover, let us refer the reader to [Section 2.2][6] for a further exposition of persistent landscapes properties.

More concretely, we will sketch how we can assume persistence landscape being functional summaries mapping persistence diagrams to Banach spaces, following the discussion in [6]. Moreover, we will define persistence landscapes norms which will be an essential domain of definition for developing our further results, experiments and tools. Persistence landscapes norms have been used as the main tool for analyzing topological time series via TDA, which will be our next topic.

Given a measure space $(\mathcal{S}, \mathcal{A}, \mu)$, we can consider the set of functions $f : \mathcal{S} \to \mathbb{R}$ defined μ-almost everywhere whose absolute value raised to the p-th power has a finite integral for $p \in [0, \infty)$, or equivalently,

$$\parallel f \parallel_p = (\int |f|^p d\mu)^{\frac{1}{p}} < \infty.$$

By the Minkowski inequality, the following holds

$$\parallel f + g \parallel_p \leq \parallel f \parallel_p + \parallel g \parallel_p,$$

with equality if and only if $f = \lambda g$ for some scalar , $\lambda \geq 0$. Hence, the set of p-the power integrable functions, together with $\| \cdot \|_p$ form a seminormed vector space with the natural operations. We will denote the later as

$$\mathcal{L}^p(\mathcal{S}) = \{f : \mathcal{S} \to \mathbb{R} | \parallel f \parallel_p < \infty\},$$

for $1 \leq p < \infty$.

In case $p = \infty$, $\mathcal{L}^\infty(\mathcal{S})$ is the set of functions bounded almost everywhere, with the essential supremum of its absolute value as a norm. Then, we can define a its corresponding normed space as follows:

$$L^p(\mathcal{S}) = \mathcal{L}^p(\mathcal{S}) / Ker(\| \cdot \|_p).$$

By Riesz-Fischer theorem it can be proved that $L^p(\mathcal{S})$ are complete too, and hence $L^p(\mathcal{S})$ is Banach. Persistence landscapes are stable with respect the L^p-norm for $1 \leq p < \infty$ [5].

Definition 6.5. *Given a persistence landscape function $\eta(k, t) = \eta_k(t) : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$, we define the* ***persitence landscape norm*** *as $\parallel \eta \parallel_p = (\sum_{k=1}^{\infty} \parallel \eta_k \parallel_p^p)^{\frac{1}{p}}$ with the product of the counting measure and the Lebesgue measure defined on $\mathbb{N} \times \mathbb{R}$.*

The set of persistence landscapes endowed with the norm defined above lies in $L^p(\mathbb{N} \times \mathbb{R})$, and hence, it is a Banach space. Thus, we can apply results from probability in Banach spaces [25]. The latter will allow us to understand persistence lanscapes as random Borel variables with values in $L^p(\mathbb{N} \times \mathbb{R})$, $\Lambda(\omega) := \eta(X(\omega)) : \Omega \to L^p(\mathbb{N} \times \mathbb{R})$ with $||\Lambda||$ being a real-valued random variable.

Apart from results discussed in 6.1, the pointwise convergence of persistence landscapes is proven in [5]. Moreover, in [10] previous results where extended, proving the uniform convergence of the average landscape. In fact, it is shown, that the empirical process $\sqrt{n}(\frac{1}{n}\sum_{i=1}^{n} \eta_i(t) - \mathbb{E}[\eta])$ for $t \in [0, T]$, converges to a

Gaussian process on $[0, T]$, as we have seen. Furthermore, a rate of convergence is established.

Additionally to persistence landscapes, and taking profit of the paragraph above, we introduce persistence silhouettes, introduced in [10].

Definition 6.6. *Consider a persistence diagram $\mathcal{D} \backslash \Delta$ such that $|\mathcal{D} \backslash \Delta| = m$, a* ***weighted silhouette*** *is defined as*

$$\mathbb{F}(D; t) = \phi(t) := \frac{\sum_{j=1}^{m} w_j \Lambda_j(t)}{\sum_{j=1}^{m} w_j}.$$

Moreover, for every $p \in (0, \infty]$ we define the ***power-weighted silhouette*** *as*

$$\mathbb{F}(D; t) = \phi^{(p)}(t) := \frac{\sum_{j=1}^{m} |d_j - b_j|^p \Lambda_j(t)}{\sum_{j=1}^{m} |d_j - b_j|^p}.$$

Intuitively speaking, when p is small, $\phi^{(p)}(t)$ is dominated by the effect of low persistence pairs. Conversely, when p is large, $\phi^{(p)}(t)$ it is dominated by the most persistent pair. Observe that the persistence silhouette functional summary maps persistence diagrams to $\mathbb{R}$-valued functions.

As stated in [10], the power-weighted silhouette preserves the condition of being one-Lipschitz. In fact, this is true for any choice of non-negative weights. Hence, the convergence results mentioned in the previous article and sections hold.

For more detail, let refer the reader to [10], as specified above.

6.3 Persistent Entropy

In addition, we introduce persistent entropy, defined in its current form in [36], and introduced first in [11], a concept based on Shannon's entropy [38] which summarizes the persistence diagrams' topological information in a single number, which sometimes may be convenient. We will present two functions appearing first in [2], as an adaptation of this concept. Moreover, we will propose a functional summary based on persistence entropy satisfying the L-Lipschitz condition.

Definition 6.7. *Let X be a discrete random variable on a probabilty space $(\Omega, \mathcal{F}, P)$with its corresponding probability mass function p_X, the* ***entropy*** *H is defined as*

$$H(X) = \mathbb{E}[-\log(p_X)].$$

In fact, let $supp(X) = \{x_1, ..., x_n\}$, the entropy can be explicitly written as

$$\sum_{i=1}^{n} p_X(x_i) \log_b(p_X(x_i)),$$

where b is the base of the logarithm. Usually $b = 2$, and its corresponding unit its called bit.

The previous concept is adapted as follows.

Definition 6.8. *Given a persistence diagram $D = \{[b_i, d_i)\}_{1 \leq i \leq n}$ with $D \in \mathcal{D}_{\mathcal{F}}$, $\mathcal{D}_{\mathcal{F}} = \{D \in \mathcal{D}$: such that $d_i < \infty$ for all $[b_i, d_i) \in D\}$, the **persistent entropy** of D, denoted by $E(D)$ is defined as:*

$$E(D) = -\sum_{i=1}^{n} \frac{l_i}{L} \log(\frac{l_i}{L})$$

where l_i is the length of each interval $[b_i, d_i)$ and log will refer to the log-base-2 function.

Persistent entropy stability is proven in [11]. To do so, they adapt Shannon's entropy stability results to the metric space of persistent diagrams, and combine them with stability results of persistent homology . They generalized the results to the set persistence diagrams $\mathcal{D}$.

In [2] they introduce two functions based on persistence entropy to introduce new uses of this concept, the entropy summary function and the normalized entropy summary function.

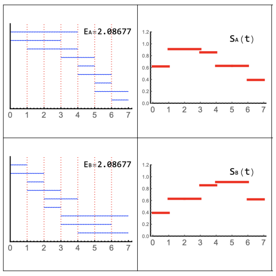


Figure 6.2: Example of two persistence barcodes with their corresponding persistence entropy and their corresponding entropy summary function

Definition 6.9. *The **entropy summary function (ES-function)** of a persistence diagram $D = \{[b_i, d_i)\}_{1 \leq i \leq n}$, with $D \in \mathcal{D}_{\mathcal{F}}$, $\mathcal{D}_{\mathcal{F}} = \{D \in \mathcal{D}$: such that $d_i < \infty$ for all $[b_i, d_i) \in D\}$ is the real-valued piecewise constant function defined as*

$$S_D(t) = -\sum_{i=1}^{n} w_i(t) \frac{l_i}{L} \log(\frac{l_i}{L})$$

where $w_i(t) = 1$ if $b_i \leq t \leq d_i$ and $w_i(t) = 0$ otherwise.

Note that the entropy summary function maps a persistence diagram and an instant t with the corresponding entropy of the intervals that are "alive" at t. Let us refer the reader to [Theorem 4.2][2] to check the stability of the ES-function.

In addition to the previous function, a scale-invariant function is presented.

Definition 6.10. *The* ***normalized entropy summary function (NES-function)*** *of a persistence diagram* $D = \{[b_i, d_i)\}_{1 \leq i \leq n}$*, with* $D \in \mathcal{D}_{\mathcal{F}}$*,* $\mathcal{D}_{\mathcal{F}} = \{D \in \mathcal{D} :$ *such that* $d_i < \infty$ *for all* $[b_i, d_i) \in D\}$ *is the real-valued piecewise constant function defined as*

$$NES_D(t) = \frac{S_D(t)}{\parallel S_D \parallel_1}$$

Let us refer the reader to [Theorem 4.4][2] to check the stability of the NES-function.

Additionally, we would like to introduce a new candidate to functional summary, satisfying the property of being L-Lipschitz. The main idea was to introduce a functional summary based on persistence entropy, that, in contrast to NES and ES functions, could satisfy the latter. Intuitively, the following summary intends to provide information regarding the quantity of persistence of persistent entropy values along t, together with the corresponding persistent entropy value generated from generators that have been born at time t or before.

Let $D = \{[b_i, d_i)\}_{1 \leq i \leq n}$ be a persistence diagram, and lets denote $NE(\cdot)$ be the persistence entropy function but this time normalized. Moreover, lets define

$$NE(t) = E(D \backslash \{p_i = (b_i, d_i) \in D : b_i > t\})$$

Lets denote $f : [0, 1] \to [0, 1]$ as monotonous increasing function.

Without loss of generality, given the total order on $\mathbb{R}^2$ given by $(b_i, d_i) < (b_j, d_j)$ if and only if $(b_i < b_j)$ or $(b_i = b_j)$ and $d_i < d_j$, suppose that point indexes in D are reorganized in ascending order leading into $D = \{(b_i, d_i)\}_{1 \leq i \leq n}$ satisfying that for all (b_i, d_i), $(b_{i-1}, d_{i-1}) < (b_i, d_i) < (b_{i+1}, d_{i+1})$.

Then, we define

$$\Lambda_{p_i, p_j}(t) = \begin{cases} (t - b_i) & t \in [b_i, \frac{b_i + b_j}{2}], j = i + 1 \\ (b_j - t) & t \in [\frac{b_i + b_j}{2}, b_j], j = i + 1 \\ 0 & otherwise \end{cases}$$

Remark 6.11. Observe that for all points in the diagram with non-consecutive birth indexes $\Lambda_{p_i, p_j}(t) = 0$. The same holds for points in the diagram with same birth value.

Finally we define our persistence entropy based functional summary candidate as

$$PEFSC(t) = \sum_{i,j} \sqrt{f(E(t))} \Lambda_{p_i, p_j}(t)$$

Remark 6.12. Observe that $PEFSC(t)$ is 1-Lipschitz. In fact, it is $max_t(\sqrt{f(E(t))})$-Lipschitz. Hence, results given in 6.1 hold.

Intuitively, the function f serves us to weigh how much we want normalized persistence entropy values to affect the function in a certain time t.

We propose the latter function for further study of its properties, stability and uses.

Chapter 7

Introduction to TDA for Financial Time Series

In a few articles such as [17][18][32][33][35], TDA has been used or discussed as a tool for studying financial time series. Two primary methodologies have arisen to embed time series into a convenient space where to be treated as point clouds (or a series of point clouds) to later compute persistence homology. In [17], M. Guidea and Y. Katz apply the sliding window method to a 4-dimensional signal consisting of four 1-dimensional time series of stock indexes. However, the latter requires multiple signals.

When the analysis has to be performed over a single time series, the standard approach [31][35][32] relies on a combination of Takens' embedding [43] (embedding process to point cloud) and the sliding window methodology (obtain a partition or sequence of point clouds). Hence, in this chapter, we will introduce both.

To understand the financial time series's fundamental behaviour, we will present some empirical results regarding financial time series, which are commonly assumed to be accurate. Those empirical results are called stylized facts.

In the previously mentioned articles, [35][32][17], persistence landscapes are used as the main persistence diagram functional summaries to perform financial time series analysis. Leaving aside the possible discussion that would be reasonable to have, so as to wonder why L_p norms of persistence landscapes are so widely used in TDA for financial time series, once this point has been reached, we will introduce the main algorithm used to compute them. Moreover, we will present a program, together with the main library's data and methodologies used to develop it, which may serve the reader as a tool to perform empirical analyses of stocks and compare TDA results with classical statistical ones.

Later on, our work focuses on an article provided by Josep Vives, [1], that can be considered as a continuation of [17]. In fact, we prove that the results reflected on the dependency relationship between persistence landscapes functional norms and variance-covariance in multivariate time series embedded via the sliding window method are valid for time series understanding them as a realization of a weakly stationary stochastic process, and assuming that the point clouds are obtained through the time-delay embedding. Moreover, using Proposition 6.1, we will observe that, by construction, silhouettes satisfy the same results. Afterwards, we will introduce the AR(1) model. The ease of setting the autocorrelation function, the variance, and the expectation that this method provides, will help us plot the previous results.

Finally, we introduce another program that has been coded to provide a useful framework to generate time series coming from standard distributions and time series models such as ARCH(1), GARCH(1,1), and the previously mentioned AR(1). Moreover, we will propose the further study of specific properties and dependencies that will be easily intuited with certain simulations that will be provided.

7.1 Takens' embedding: From time series to point clouds

We will aim to subtract topological information from time series through persistence homology. To do so, Takens' embedding, usually known as time delay embedding, has become one of the most popular and useful computational tools. It serves us as a methodology to transform time series into point clouds. The main sources followed have been [29][43].

Briefly speaking, a discrete dynamical system (M, ϕ) consists of a compact manifold M containing the states of the system and a diffeomorphism $\phi : M \to M$ which determines the evolution of the state over time. We call M the state space, and ϕ the dynamical map. If a system (M, ϕ) is in state $x \in M$ at some time t, then $\phi(x)$ is the state at time t + 1. Hence, the trajectory of the space is determined by ϕ. A time series can be assumed to be a series of observations of a dynamical system trajectory, and therefore, a common goal is to reconstruct the main rules of the dynamical system from these observations. We call $f : M \to \mathbb{R}$ an observable. Takens' theorem, provides certain conditions under which a smooth attractor (a set of numerical values towards which a system tends to involve) can be reconstructed from a set of observed values. The latter is formalized as follows in [43].

Theorem 7.1. *Let M be a manifold of dimension d. For pairs (ϕ, f), where $\phi : M \to M$ is a smooth diffeomorphism and $f : M \to \mathbb{R}$, it is a generic property that the $(2d+1)$ delay observation map given by $\Phi : M \to \mathbb{R}^{2d+1}$ given by $\Phi_{(\phi,f)}(x) = (f(x), f \circ \phi(x), ..., f \circ \phi^{2d}(x))$ is an embedding.*

In practice, ϕ can be assumed to depend on a fixed τ which will be called the delay parameter, in the sense that, if x is the state at time t, then $\phi_\tau(x)$ is the state x at time τ. Takens proved that the if $m > 2d$ then ϕ is generally a faithful embedding.

Hence, suppose we have a time series $X = \{x_1, ..., x_N\}$, according to Takens' embedding theorem [43], the pair (m, τ), characterizes the entire transformation, leading into a trajectory matrix $f(X) \in \mathbb{R}^{(N-(m-1)\tau)\times m}$ defined as

$$f(X) = \begin{bmatrix} X_{(m-1)\tau} \\ X_{1+(m-1)\tau} \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} x_0 & x_\tau & \cdot\ \cdot\ \cdot & x_{(m-1)\tau} \\ x_\tau & x_{1+\tau} & \cdot\ \cdot\ \cdot & x_{1+(m-1)\tau} \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ x_{N-(m-1)\tau} & x_{N-(m-2)\tau} & \cdot\ \cdot\ \cdot & x_N \end{bmatrix}$$

In fact, we are generating the m most recent equally spaced points up to time t with a predefined gap τ. Notice that the choice of parameters m and τ will be crucial.

In [31], they combined the time delay embedding with the sliding window method to obtain set of point clouds from a single time series. In our initial practical implementation, we will use such methodology, following the first algorithm proposed in [18].

7.2 Sliding window of time series

It will be necessary for time series analysis to decide whether or not the analysis has to be done over the whole time series, or segment by segment. The sliding window method is defined as follows:

Suppose we have a time series $X = \{x_1, ..., x_n\}$, then, the sliding window methods divides the time series into partitions $SW_{v,g}(t) = \{x_t, x_{t+g} ..., x_{t+vg}\}$ where v denotes the size of the window and g denotes the time gap. In our work, we will suppose $g = 1$ in this particular method. Hence, in this case, the value v will be the crucial parameter.

Notably, in article [17], M. Guidea and Y. Katz, applied the sliding window method on a 4-dimensional time series composed of four US stock indexes, to later perform a functional norms persistence landscapes analysis.

7.3 Financial Time Series and Stylized Facts

In this section, we briefly introduce some statistical properties that financial time series tend to share to better understand the subject of matter. Those empirical findings, consistent across markets, are called stylized facts. We refer the reader to [37] for further information and discussion. Let us suppose that the financial time series follows a stochastic process $\{X_t, t \in \mathbb{N}\}$.

- Non-stationarity: Financial time series tend to be non-stationary. In fact, $\mathbb{E}[X_t]$ usually follows a non-linear trend. One common approach is to transform our time series into $\{R_t, t \in \mathbb{N}\}$ where $R_t = \log(\frac{X_t}{X_{t-1}}) = \log(X_t) - \log(X_{t-1})$ and assume stationarity.

- Dependence: The autocorrelation function of the returns process $\{R_t, t \in \mathbb{N}\}$ is largely insignificant, i.e, $\rho(l) \approx 0$ for all $l > 0$ except from very small intraday time scales. However, $\mathbb{C}[R_t^2, R_{t+l}^2]$ and $\mathbb{C}[|R_t|, |R_{t+l}|]$ do not need exhibit the same property.

- Distribution: The empiririical kurtosis is greater than 3, which is the kurtosis associated to the standard normal Gaussian distribution. This implies a bigger density of return values near the mean, and in the tails, leaving intermediate values with less density, compared to the standard normal Gaussian distribution. In fact, it is observed and increasingly positive kurtosis as the time interval decreases.

- Intermittency/Volatility clustering: Periods of big movements are followed by periods of downward movements and vice versa. The latter is known as conditional heteroskedasticity.

7.4 Practical implementation on Financial Time Series

As we have mentioned, one of the most extended financial time series analysis using TDA methodologies, is based on computing persistence landscapes L_p norms on point clouds obtained through a combination of Takens' embedding and the sliding window approach.

To illustrate this point, in [35], they developed a TDA-portfolio method for cryptocurrencies. Moreover, in [18], they intended to develop a TDA based enhanced indexing algorithm.

First, in this section we introduce the latter algorithm. Afterwards, we will give some explanatory comments regarding the code that we have developed to perform such computations. Moreover, we will provide some empirical results.

Algorithm 1: Generating persistence landscapes Lp norm time series

1. Transform the in-sample index data into log returns
   $x_i = log(P_t) - log(P_{t-1})$ with P_t corresponding to price at time t.

2. Apply Takens' embedding to obtain a $((N - (m - 1)\tau) \times m)$ matrix

$$f(X) = \begin{bmatrix} X_{(m-1)\tau} \\ X_{1+(m-1)\tau} \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} x_0 & x_\tau & \cdot\ \cdot\ \cdot & x_{(m-1)\tau} \\ x_\tau & x_{1+\tau} & \cdot\ \cdot\ \cdot & x_{1+(m-1)\tau} \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ x_{N-(m-1)\tau} & x_{N-(m-2)\tau} & \cdot\ \cdot\ \cdot & x_N \end{bmatrix}$$

3. Apply sliding window method with time window size v in order to obtain $N - (v + 1)$ point clouds $\{PC_i\}_{1 \leq i \leq N-(v+1)}$of the form

$$PC_i = \begin{bmatrix} x_i & x_{i+\tau} & \cdot\ \cdot\ \cdot & x_{i+(m-1)\tau} \\ x_{i+1} & x_{(i+1)+\tau} & \cdot\ \cdot\ \cdot & x_{(i+1)+(m-1)\tau} \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ x_{(i+v)-(m-1)\tau} & x_{(i+v)-(m-2)\tau} & \cdot\ \cdot\ \cdot & x_{(i+v)} \end{bmatrix}$$

4. Compute persistence diagram from Vietoris-Rips reconstruction and its corresponding persistence landscape and norm.

7.4.1 Developed tools, Methodology and Data

In FinTDA.ipybn A.1, we have developed a class (IndexTDA) with which several empirical results can be easily computed from stock indexes in a selectable date range. First of all, as we have mentioned, for a given v, referring to the window size of the sliding window method, and parameters τ (Takens' Embedding time delay parameter) and m (Takens' Embedding dimension parameter) one can obtain the persistence landscape L_p norm time series from the corresponding generated point clouds mentioned in the algorithm above. Moreover, persistence entropy can be computed by means of the persim python's package. In fact, for that given v, the user can easily compute/plot the standard deviation, the kurtosis, the autocorrelation for $l = 1$, the value at risk (VaR computed by bootstrapping) and the expected shortfall (ES) for the whole generated point clouds, and obviously the index values and their corresponding logarithmic returns. We urge the reader to use those tools to understand and empirically test the behaviour of persistence landscapes L_p norm.

To compute Vietoris-Rips complexes from point cloud data together with their corresponding persistence diagrams, we have used the ripser.py package [44]. We decided using ripser, since, as it is specified, it outperforms other well-known packages such as GUDHI by a factor of more than 40 in computation time and a factor of more than 15 in memory efficiency. On the other hand, persistence landscapes have been computed employing the Landscape class from GUDHI package. In the same way, to compute the Takens' Embedding, we relied on its GUDHI class TimeDelayEmbedding. Moreover, we compute the Lp norms utilizing skfda python's package. Furthermore, to compute the fourth empirical moment, the standard deviation and the autocorrelation we relied on the scipy.stats package, NumPy package and statsmodels.tsa.API respectively.

On the other hand, data and dates are managed via pandas, and DateTime packages, respectively. Stock indexes data has been extracted from Yahoo Finance. In fact, in */Code*, we provide historical data from SP500, SP100, NASDAQ and VIX that can be easily used in our IndexTDA class. Moreover, we have developed the rolling window strategy introduced in [18] in order to perform a more decoupled analysis of the stock indexes.

Let us refer the reader to A.2 to check some results obtained from the previous implementations.

Comments regarding the selection of the embedding parameter

As mentioned in 7.1, parameters (τ, m) fully characterize the embedding. Hence, we will provide some intuition for parameter fitting.

First of all, several methods regarding the choice of parameter τ have been developed. For instance, one of the most extended methods relies on the autocorrelation function, setting $\tau = l$ being l the smallest lag value where the autocorrelation function becomes insignificant [32][23]. As mentioned in 7.3, the autocorrelation function for returns time series remains proper to 0 for lags $l > 0$. Hence, under this context, τ is commonly set to 1, $\tau = 1$ [32][35][18].

Secondly, regarding parameter m, in [32] they used the false nearest neighbour and defined m as the smallest integer such that the nearest neighbours of each point in dimension m remain nearest neighbours in dimension $d + 1$. However, some author assume m to be equal to two or three [18].

Another crucial parameter about which there is not much literature is the parameter v. In fact, in [35] and [18] they simply assume it to be 30 or 21 for index analysis.

7.4.2 A computed example

Here we introduce the result of computing Algorithm 1 on the SP500 index. As previously mentioned, we will set $\tau = 1$, and assume $m = 2$. We will choose the window size value $v = 63$ [18], meaning that 63 different trading days will be considered in each point cloud.

First, we plot the index and its corresponding logarithmic returns:

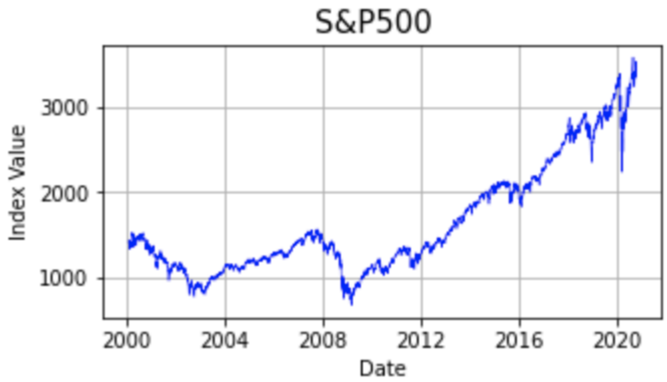


Figure 7.1: SP500 index from 01-2000 to 10-2020

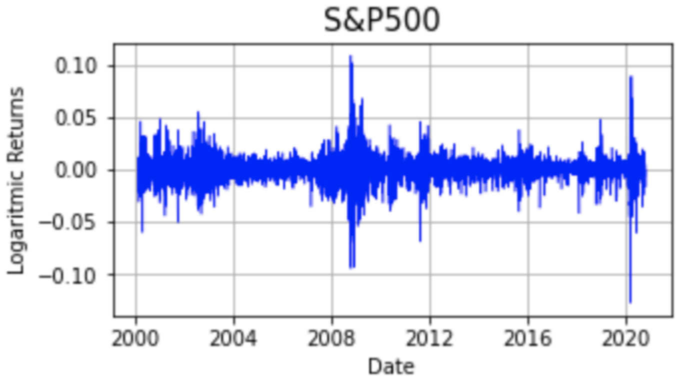


Figure 7.2: SP500 logarithmic returns from 01-2000 to 10-2020

Spearman coefficient between the standard deviation and the L_1persistence landscapes norms for $v = 21 : \rho = 0.806$.

Standard deviation and L_1 time series present a highly significant Spearman coefficient. This will motivate us to adapt the results given in [1] to our context, demonstrating a dependence on the variance and the autocorrelation function. As previously discussed in 7.3, the autocorrelation at any lag greater than 0 of a

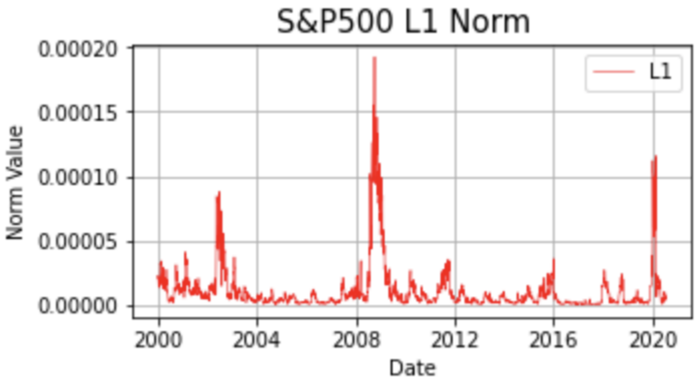


Figure 7.3: SP500 L1 Noms, $d = 2$,$v = 63$ from 01-2000 to 10-2020

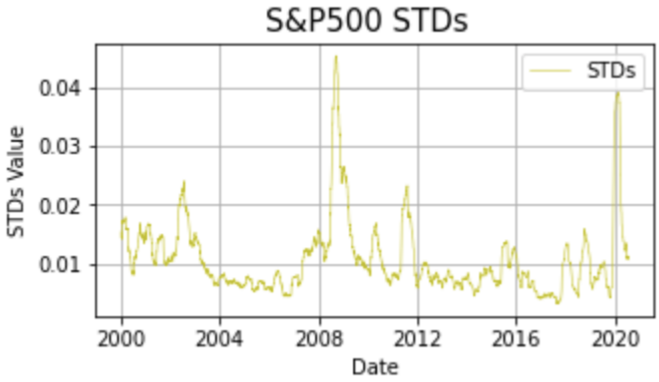


Figure 7.4: SP500 L1 Noms, $d = 2$,$v = 63$ from 01-2000 to 10-2020

financial time series is near to 0. Therefore, as we will show, it will not affect the L_1 norm series. We refer the reader to A.2 for checking the window's size influence v on the obtained Spearman coefficient. We will observe that larger windows reflect a greater correlation, meaning that the autocorrelation's nullity becomes more significant.

7.5 Adapting theoretical dependency results, equivalences and further work proposals

In this section we adapt the theoretical results given in [1] in order to prove their validity under construction given in Algorithm 1, assuming that our time series is a realization of a weakly stationary stochastic process.

Proposition 7.2. *Suppose that we have an $\mathbb{R}$-valued well-defined second order stochastic process $\{X_t, t \in \mathbb{N}\}$. Then, for a certain $t \geq 0, \tau \geq 1$,$m > 1$ we define the following random vectors $X_1 = (X_t, X_{t+\tau}, ..., X_{t+(m-1)\tau})$,$X_2 = (X_{t+\tau}, X_{t+2\tau}, ..., X_{t+m\tau})$, ...,$X_N = (X_{t+(N-1)\tau}, X_{t+N\tau}, ..., X_{t+(N+m-2)\tau})$ such that $\mathbb{X} = (X_1, ..., X_N)$ describes an N point data set in $\mathbb{R}^m$. Lets denote X_j^i to the random variable allocated in the j-th component of the i-th random vector, then*

$$\mathbb{E}[\| \eta \|_p] \leq C(N) \cdot \sum_{i=1}^{N} (\sum_{i=1}^{m} \mathbb{V}[X_j^i])^{\frac{p+1}{2p}} < \infty \tag{7.1}$$

where $\| \eta \|_p$ denotes the p-norm of the persistence landscape η, and $C(N)$ is the number of non zero k-persistence landscapes for any k.

Proof. Since $\{X_t, t \in \mathbb{N}\}$ is a well-defined second-order stochastic process, from 5.4, we have that $\mathbb{E}[X_t], \mathbb{V}[X_t] < \infty$ for all $t \in \mathbb{N}$. Therefore, the proof is analogous to [Proposition 4][1]. □

Corollary 7.3. *Let $\{X_t, t \in \mathbb{N}\}$ be a well-defined second order weakly stationary stochastic process and suppose that $\mathbb{X} = (X_1, ..., X_N)$ is constructed as in Proposition 7.2 then, under the same notation*

$$\mathbb{E}[\| \eta \|_p] \leq C(N) \cdot N \cdot (m \cdot \mathbb{V}[X_t])^{\frac{p+1}{2p}} < \infty. \tag{7.2}$$

Proof. Suppose that we have the same construction as in Proposition 7.2. Let $\{X_t, t \in \mathbb{N}\}$ be second order weakly stochastic process, then it satisfies that $\mathbb{V}[X_k]$ is constant for all $k \in \mathbb{N}$. Hence, without loss of generality, we can denote $\mathbb{V}[X_t]$ to be the variance for each random value in the process. Therefore, from Equation (7.1) we obtain Equation (7.2). □

Remark 7.4. Let $\{X_t, t \in \mathbb{N}\}$ be a well-defined second order weakly stationary stochastic process, and let $\mathbb{X} = (X_1, ..., X_N)$ be constructed as in Proposition 7.2. Then, implications of [Proposition 5][Theorem 6] from [1] hold.

Proof. Let $\{X_t, t \in \mathbb{N}\}$ be covariance stationary. Then, for a fixed τ, following the construction given in Proposition 7.2, expectations and variance-covariance matrices remain equal for all $X_1, ..., X_N$. Hence, [Proposition 5][Theorem 6] from [1] can be applied. □

Theorem 7.5. $\{X_t, t \in \mathbb{N}\}$ *be a well-defined second order weakly stationary stochastic process. For* $m = 2$ *and fixed* N *and* τ*, consider the associated point cloud* $\mathbb{X} = (X_1, ..., X_N)$ *and the corresponding lanscape* η*. Then we have*

$$\mathbb{E}[\| \eta \|_1] \leq C(N) \cdot L_1(N) \cdot N \cdot \mathbb{V}[X_t](1 - |\rho(\tau)|) \leq 4^N \cdot N \cdot \mathbb{V}[X_t](1 - |\rho(\tau)|). \quad (7.3)$$

where $C(N)$ *is the number of non zero k-persistence landscapes for any k and* $L_1(N)$ *denotes the number of 1D loops described by an N point data set.*

Proof. The proof is analogous to [Theorem 7][1]. The main difference lies in the 2x2 variance covariance matrix Σ. Under our construction and assuming covariance stationarity, we have that, for each $\{X_i\}_{1 \leq i \leq N}$ Σ can be described as follows.

$$\Sigma = \begin{bmatrix} \mathbb{V}[X_t] & \rho(\tau)\mathbb{V}[X_t] \\ \rho(\tau)\mathbb{V}[X_t] & \mathbb{V}[X_t] \end{bmatrix}$$

Therefore, if we diagonalize Σ, under the notation of [1] we obtain the following eigenvalues:

$$\theta_1 = \mathbb{V}[X_t](1 + |\rho(\tau)|)$$
$$\theta_2 = \mathbb{V}[X_t](1 - |\rho(\tau)|)$$

□

Remark 7.6. We use the absolute value of the autocorrelation to ensure that $\theta_2 \leq \theta_1$.

Remark 7.7. As it appears in [Remark 8][1], observe that if $|\rho(\tau)| \to 1$, then $\mathbb{E}[\| \eta \|_1] \to 0$. Contrariwise, if $p(\tau) = 0$, then 7.3 is only characterized from the effect of $\mathbb{V}[X_t]$.

7.5.1 Validity of the results for Silhouettes

Let $\mathcal{D} = \{p_i(b_i, d_i)_{i \in I}\}$ be a persistence diagram and let $\Lambda_p(t)$ be the triangle functions defined in 6.6. Moreover, recall from 6.6 that a persistence silhouette is the functional summary $\mathbb{F} : \mathcal{D} \to \mathcal{F}_1$ defined as follows:

$$\mathbb{F}(D; t) = \phi(t) := \frac{\sum_{j=1}^{|D|} w_j \Lambda_j(t)}{\sum_{j=1}^{|D|} w_j}$$

Since persistence silhouettes are defined as a weighed average of $\Lambda_p(t)$, one can make the following two observations:

1. Silhouettes are one-Lipschitz functions. Thus, Proposition 6.1 holds.

2. $\phi(t) \leq \eta(1, t)$. Then $\| \phi(t) \|_p^p \leq \| \eta(1, t) \|_p^p$

Therefore, all the results given above and in [1] hold.

In particular, for $\mathbb{E}[\| \phi \|]_p$ the boundaries given in [Proposition 4][1], Proposition 7.2 and in Corollary 7.3 would remain equal but suppressing the term $C(N)$. The same difference would appear for the boundary of $\mathbb{E}[\| \phi \|_1]$ given in [Theorem 7][1] and in 7.5. In fact, notice that any collection of L-Lipschitz functional summaries bounded by $\eta(1, t)$ would satisfy all the results provided in [1].

7.5.2 Further work and intuition

Results given in [1], and their analogous previously presented, assume a fixed number of points N in the point cloud $\mathbb{X} = (X_1, ..., X_N) \in \mathbb{R}^d$. However, one might observe that this parameter is crucial for the meaning of our analysis, as it is mentioned in 7.2. Observe that under our construction, $N = v$. As one may notice, it will not be the same to perform the analysis on a point cloud of cardinality 10, 100, 1000 or in any arbitrary window size. At the cardinality of the point cloud increases, the density of the point cloud will increase which might lead into less persistent features. Therefore, we propose further study of the persistence homology dependence with respect to the cardinality of the point cloud to be analyzed. In particular, we propose studying the second, third, and fourth moments effect on the evolution of the latter dependency.

We plot, under our construction, different embedded time series, coming from known distributions with different kurtosis values, to show that possible dependence.

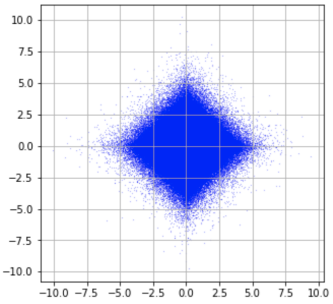


Figure 7.5: Laplace i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 3$ for $N = 10000000$.

We refer the reader to B.2 to visualize the same results for a larger set of distributions.

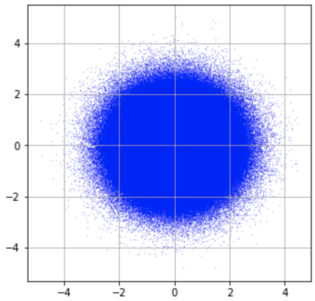


Figure 7.6: Normal i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 0$ for $N = 10000000$.

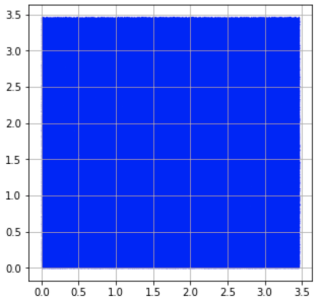


Figure 7.7: Uniform i.i.d noise with $\mu = 0, \sigma = 1, \kappa = -1.2$ for $N = 10000000$.

7.5.3 The AR(1) model

The intention in this subsection is to introduce the AR(1) model, the simplest model of the AR(p) model family, a subfamily of the family of ARMA(p,q) models (autoregressive and moving average models) [45]. It will provide us with enormous ease when it comes to fit expectation, variance and autocorrelation function parameters of the model. Thus, that will serve us to plot an exemplification of the above results. This model's main intuition is that the observed time series depends on a weighted linear sum of the p past values (in AR(1), p=1), and a random shock. Thus the name "autoregressive" derives from this idea.

Definition 7.8. *Let $Z = \{Z_j, j \in \mathbb{Z}\}$ be a centered white noise with variance $\sigma^2 > 0$.*

$$Z \sim WN(0, \sigma^2)$$

We say that $Y = \{Y_j, j \in \mathbb{Z}\}$ is an AR(1) process if it is a stationary process and satisfies the equation

$$Y_j = \phi Y_{j-1} + Z_j,\ j \in \mathbb{Z},\ \phi \in (-1, 1) \tag{7.4}$$

From the previous definition, we can observe the following by iteration.

$$\begin{aligned} Y_j &= \phi(\phi Y_{j-2} + Z_{j-1}) + Z_j \\ &= \phi^2 Y_{j-2} + \phi Z_{j-1} + Z_j \\ &= ... \\ &= Z_j + \phi Z_{j-1} + \phi^2 Z_{j-2} + ... + \phi^k Z_{j-k} + \phi^{k+1} Y_{j-k-1} \\ &= \sum_{i=0}^{k} \phi^i Z_{j-i} \ + \ \phi^{k+1} Y_{j-k-1} \end{aligned}$$

Since $\phi^i Z_{j-i}$ are centered and pairwise uncorrelated square integrable random variables, applying [Corollary 2.4][45] we have that

$$\sum_{i=0}^{\infty} \phi^i Z_{j-1} < \infty(L^2) \iff \sum_{i=0}^{\infty} \mathbb{V}[\phi^i Z_{j-1}] = \sum_{i=0}^{\infty} \phi^{2i}\sigma^2 = \frac{\sigma^2}{1-\phi^2} < \infty. \tag{7.5}$$

Hence, if $|\phi| < 1$ we can conclude that

$$Y_j = \sum_{i=0}^{\infty} \phi^i Z_{j-1} \ (L^2) \tag{7.6}$$

Notice that

$$\mathbb{E}|Y_j - \sum_{i=0}^{\infty} \phi^i Z_{j-1}|^2 = \phi^{2(k+1)} \mathbb{E}[Y_{j-k-1}^2] \xrightarrow{k\to\infty} 0 \tag{7.7}$$

Observe that stationarity implies $\mathbb{E}[Y_{j-k-1}^2]$ to be constant and $|\phi|^{2(k+1)}$ to 0 when $k \to \infty$.

Hence, for any $j \in \mathbb{Z}$ we have

1. $\mathbb{E}[Y_j] = 0$
2. $\mathbb{V}[Y_j] = \frac{\sigma^2}{1-\phi^2} > 0$
3. $\mathbb{C}[Y_j, Y_{j+l}] = \frac{\sigma^2\phi^l}{1-\sigma^2}$
4. $\rho(l) = \phi^l$

Remark 7.9. Observe $Y_j = c + \phi Y_{j-1} + Z$ with $Z \sim WN(0, \sigma^2)$ is an AR(1) process too. In this particular case, $\mathbb{E}[Y_j] = \frac{c}{1-\phi}$

Therefore, we can force an AR(1) process to have a certain μ, σ, $\rho(1)$ by defining it as:

$$Y_j = \mu \cdot (1 - \rho(1)) + \rho(1) Y_{j-1} + Z_j, \;\; Z \sim WN\Big(0, \sqrt{\sigma^2 \cdot (1 - \rho(1)^2)}\,\Big) \tag{7.8}$$

We use 7.8 to generate multiple time series simulations of length N for a range of fixed values of variance and autocorrelation to compute the sampled mean of the corresponding persistence landscape L_1 norms for each of the pairs variance-autocorrelation. We aim to visualize variance and autocorrelation's dependency effects on functional persistence landscapes L_1 norms to provide an snapshot of the results presented above. To generate the following plot, we have used the code presented in B.1. Particularly we have used the method

std_corr_persistence_landscape_lp_norm_surface.

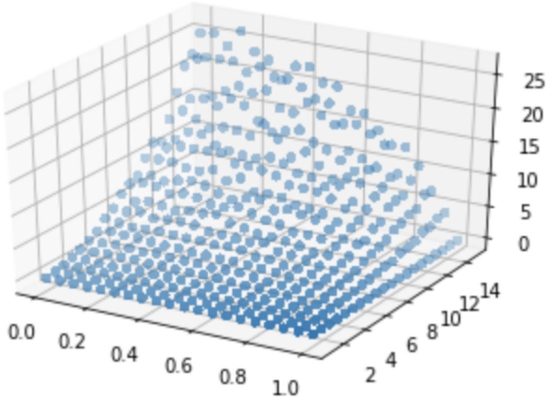


Figure 7.8: We have used the AR (1) model to generate 750 time series, with 200 values each, for fixed equidistant values of standard deviation and autocorrelation within a given range. In particular, we have used the default values that appear in the provided code. Computing the L_1 norm of the associated persistence landscape with each of these 750 simulations, we obtain an estimated value. In this graph, we can see show the effect of the results previously presented. We can see that, as the autocorrelation value increases from 0 to 1, the value of the norm L_1 is nullified. The opposite occurs with the deviation's value.

7.5.4 Developed tools

In EmpTDA.ipynb B.1, we have developed a set of python tools that can be very useful to extract topological information from simulated time series. On the one hand, we provide a bunch of methods to simulate time series using some of the most popular stochastic models, such as AR(1), ARCH(1) and GARCH(1,1).

We provide methods to simulate time series from the Laplace, logistic, normal, semicircular and uniform distributions. Moreover, we provide the necessary tools to plot the simulated time series together with their autocorrelation and partial autocorrelation functions. On the other hand, relying on the same python packages mentioned in 7.4.1, we provide the following TDA methods in the class TDATimeSeries. First, we dispense the tools to compute Takens' embedding and plot the embedded time series. Second, we give the methods to compute and plot persistence diagrams and their corresponding density. Moreover, we provide the tools to compute and plot the simulated time series's persistence landscape, together with its corresponding norm. Furthermore, power weighted silhouettes, power weighted silhouettes L_p norms, persistence entropy, ES and NES functions, are implemented. We encourage the reader to use the presented tools in order to perform his experiments.

Chapter 8

Conclusions

The early motivation for this work arose after hypothesizing that the data's topological structure could be relevant when it comes to providing alternative information to the more classical methods. It did not take long to discover a branch dedicated to this topic, a point of confluence between Algebraic Topology, Statistics and Computation Theory, Topological Data Analysis.

In this work, as the reader must have verified, we provide the necessary theoretical bases for a first approach to one of the main tools of topological data analysis, persistent homology, which will allow us to elucidate the most significant topological characteristics of data's underlying topological space. Later, we provide a framework to understand homological persistence from a statistical point of view. This framework is made up of a set of maps that, under certain assumptions, will allow us to obtain a more conductive stable and reliable representation for data analysis.

Under this same framework, we present the persistence landscapes and silhouettes. In addition, we present a new function that could serve as a descriptor, based on persistence entropy. As future work, we propose a more detailed study of this function, as well as its possible uses (if any), behaviour and stability.

Finally, we focus on the topological analysis of time series, with particular emphasis on financial time series, for which we can find a series of articles [17][35][18][32] among others.

As is well known by now, in the articles cited above, two types of methods are discerned to obtain point clouds, where persistent homology and its corresponding descriptors can be calculated.

Taking advantage of an article provided by Josep Vives [1], we demonstrate the equivalence of his results, especially thought as a continuation of [17], for univariate time series conceived as realizations of weakly stationary stochastic processes. In addition, we affirm the validity of the same for silhouettes, and we empirically show the dependency results obtained through simulations AR (1) model.

In this same work, we developed a series of Python tools, which can be highly useful for anyone who wants to work with TDA for time series. These tools will allow us to calculate, simulate and visualize different results, both related to TDA tools and classic data analysis.

In conclusion, we would like to add that there is much work to be done in this field. In particular, we propose the study of the dependence of homological persistence with respect to the cardinality of the set of points to be analyzed under the construction of [1] and the one provided in this work. Furthermore, we urge to study the effect of third and fourth-order moments under this type of construction and its possible influence on the effect produced by the change of cardinality in the set of points.

Appendix A

TDA Financial Time Series

A.1 Code

Listing A.1: Insert code directly in your document

```
import pandas as pd
import numpy as np
import gudhi as gd
import statsmodels.api as sm
import gudhi.representations
from gudhi.point_cloud.timedelay import TimeDelayEmbedding
from gudhi.representations.vector_methods import Landscape
import matplotlib.pyplot as plt
from ripser import ripser
from persim import plot_diagrams
from numpy import linalg as LA
import datetime as dt
import skfda.misc.metrics as mt
import skfda
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import spearmanr,kurtosis,norm
import random
import persim.persistent_entropy as pe
import statsmodels.tsa.api as smt


nasdaq = 'NASDAQ1971+.csv'
sp500 = 'S&P500.csv'
sp100Global = 'S&PGLOBAL100.csv'
vix = 'VIX.csv'

class IndexTDA():
    '''
    filename:= [string] Filename of the csv we want to perform the analysis
    startDate:= [(year,month,day)]The starting date from which we want to
    perform our analysis
    d:= [int]Taken's Embedding dimension parameter
    tau:= [int] Time delay parameter
    v := [int] Number of days to take into consideration for each pointcloud
    d1_d1 := [Array[(int,int)] Array of tuples with rolling windows dimensions
    '''
    def __init__(self,filename,startDate=None,endDate=None,d=2,tau =1,skip=1
    ,v=21,d1_d2 = [(126,63),(126, 42),(126, 21),(63, 21)], p = 1,
    kinConsideration = 1000, res = 1000):

        self.filename = filename
        self.data = pd.read_csv(self.filename)
        self.data['Date'] = pd.to_datetime(self.data['Date'])
        self.data.set_index('Date', inplace = True)
        self.data['Close'] = [float(c) for c in self.data.Close]
        self.data['LogRet'] = np.log(self.data.Close) - np.log(self.data.Close.shift(1))
        self.d1_d2 = d1_d2
        aux = self.data.index[1]
        firstday = (aux.year,aux.month,aux.day)
        self.startDate = firstday if (not startDate) else startDate
        self.startIndex = self.getIndex(self.startDate)
        aux = self.data.index[-1]
        lastday = (aux.year,aux.month,aux.day)
        self.endDate = lastday if (not endDate) else endDate
        self.finalIndex = self.getIndex(self.endDate)
        self.v = v
        self.numberOfPoints = (v - (d - 1)*tau)
        TD = TimeDelayEmbedding(dim = d,delay = tau, skip= skip)
        #ROLLING WINDOW STRATEGY POINT CLOUDS
        #===================================
        self.numberOfAnalyses = len(self.d1_d2)
        self.finalIndexes = [((self.finalIndex - self.startIndex - pair[0])
        //pair[1])*pair[1] + pair[0] for pair in d1_d2]
        self.analyses = [[self.data.LogRet[self.startIndex +
        i*pair[1]:self.startIndex + i*pair[1] + pair[0]]
        for i in range(0,(self.finalIndex - self.startIndex - pair[0])//pair[1])]
        for pair in d1_d2]
        self.TDEmbAnalyses = [[TD.__call__(window) for window in analysis]
        for analysis in self.analyses]
        self.rollingwindowLogRets = [[[window[i:self.v+i]
                    for i in range(0, len(window) - (self.v + 1))]
                        for window in analysis]
                            for analysis in self.analyses]
        self.rollingWindowPointClouds = [[[window[i:self.numberOfPoints+i]
                    for i in range(0, len(window) - (self.numberOfPoints + 1))]
```

```
                        for window in analysis]
                            for analysis in self.TDEmbAnalyses]
        #========================================
        #NON ROLLING WINDOW STRATEGY
        #===========================
        embeddedData = TD.__call__(self.data.LogRet[self.startIndex:
        self.finalIndex +1])
        self.pointClouds = [embeddedData[i:self.numberOfPoints+i] f
        or i in range(0, len(embeddedData) - (self.numberOfPoints + 1))]
        #===========================
        self.LpTS = pd.DataFrame(data=[])
        self.RWLpTS = None
        self.STDTS = pd.DataFrame(data=[])
        self.RWSTDTS = None
        self.ACFTS = pd.DataFrame(data=[])
        self.RWACFTS = None
        self.KURTS = pd.DataFrame(data=[])
        self.RWKURTS = None
        self.PETS = pd.DataFrame(data=[])
        self.VaRTS = pd.DataFrame(data=[])
        self.ExpShtfllTS = pd.DataFrame(data=[])

    def getIndex(self, dateTuple):
        return self.data.index.get_loc(dt.datetime(dateTuple[0],dateTuple[1],
        dateTuple[2]))

    def getClosePrice(self):
        return pd.Series(data = self.data.Close[self.startIndex+self.v:self.finalIndex])

    def getLogRet(self):
        return pd.Series(data = self.data.LogRet[self.startIndex+self.v:self.finalIndex])

    def getSTDTimeSeries(self):
        if self.STDTS.empty:
             self.STDTS = pd.DataFrame(data = [(self.data.LogRet[i:self.v+i].values).std()
             for i in range(self.startIndex,self.finalIndex - self.v)],
             index = self.data.index[self.startIndex:self.finalIndex - self.v])
        return self.STDTS

    def getACFTimeSeries(self):
        if self.ACFTS.empty:
            self.ACFTS = pd.DataFrame(data = [smt.acf(
            self.data.LogRet[i:self.v+i].values)[1]
            for i in range(self.startIndex,self.finalIndex - self.v)],
            index = self.data.index[self.startIndex:self.finalIndex - self.v])
        return self.ACFTS

    def getKurtosisTimeSeries(self):
```

```
    if self.KURTS.empty:
        self.KURTS = pd.DataFrame(data =
        [kurtosis(self.data.LogRet[i:self.v+i].values)
        for i in range(self.startIndex,self.finalIndex - self.v)],
        index = self.data.index[self.startIndex:self.finalIndex - self.v])
    return self.KURTS

def getlandscapesLpNormTimeSeries(self):
    if self.LpTS.empty:
        self.landscapesLpNormTimeSeries()
    return self.LpTS

def getpersistenceEntropyTimeSeries(self):
    if self.PETS.empty:
        self.persistenceEntropyTimeSeries()
    return self.PETS

def getVaRTimeSeries(self):
    if self.VaRTS.empty:
        self.VaRTS = self.VaRTimeSeries()
    return self.VaRTS

def getESTimeSeries(self):
    if self.ExpShtfllTS.empty:
        self.ExpShtfllTS = self.ESTimeSeries()
    return self.ExpShtfllTS

def getrollingWindowLandscapesLpNormTimeSeries(self):
    if self.RWLpTS is None:
        self.rollingWindowPersistenceLandscapesLpNormTimeSeries()
    return self.RWLpTS

def getrollingWindowSTDTimeSeries(self):
    if self.RWSTDTS is None:
        self.RWSTDTS = [[[lrets.values.std() for lrets in window]
        for window in analysis]
        for analysis in self.rollingwindowLogRets]
    return self.RWSTDTS

def getrollingWindowACFTimeSeries(self):
    if self.RWACFTS is None:
        self.RWACFTS = [[[smt.acf(lrets.values)[1] for lrets in window]
        for window in analysis]
        for analysis in self.rollingwindowLogRets]
    return self.RWACFTS

def getrollingWindowKurtosisTimeSeries(self):
    if self.RWKURTS is None:
            self.RWKURTS = [[[kurtosis(lrets.values) for lrets in window]
            for window in analysis] for analysis
            in self.rollingwindowLogRets]
        return self.RWKURTS

    def rollingWindowPersistenceLandscapesLpNormTimeSeries(self,p=1,
    kinConsideration = 5, res = 1000):
        rollingWindowPLLpN = []
        for analysis in self.rollingWindowPointClouds:
            wsPLLpN = []
            for window in analysis:
                wPLLpN = []
                for pc in window:
                    LS = gd.representations.Landscape(resolution=res,
                    num_landscapes = kinConsideration)
                    try:
                        diagram = ripser(pc)['dgms']
                        landscape = LS.fit_transform([diagram[1]])[0]
                        sample_range = LS.sample_range
                        if p == 1:
                            wPLLpN.append(self.L1(landscape,
                            sample_range,kinConsideration,res))
                        else:
                            wPLLpN.append(self.Lp(l
                            andscape,p,sample_range,kinConsideration,res))
                    except:
                        wPLLpN.append(0)
                wsPLLpN.append(np.array(wPLLpN))
            rollingWindowPLLpN.append(np.array(wsPLLpN))
        self.RWLpTS = rollingWindowPLLpN
        return self.RWLpTS

    '''
    Computes persistence landscapes lp norm time series
    p: [int] Lp norm value
    kinconsideration:[int] Number of landscapes into consideration
    res: [int] Number of persistence landscape function points into consideration
    '''
    def landscapesLpNormTimeSeries(self, p = 1, kinConsideration = 5, res = 1000):
        if p < 1:
            return 'p␣must␣be␣greater␣or␣equal␣than␣1'
        LS = gd.representations.Landscape
        (resolution=res,
        num_landscapes = kinConsideration)
        Lp = []
        for pc in self.pointClouds:
            try:
                diagram = ripser(pc)['dgms']
                landscape = LS.fit_transform([diagram[1]])[0]
                sample_range = LS.sample_range
                if p == 1:
                    Lp.append(self.L1(landscape,sample_range,
                    kinConsideration,res))
                else:
                    Lp.append(self.Lp(landscape,p,sample_range,
                    kinConsideration,res))
            except:
                Lp.append(0)
        self.LpTS = pd.DataFrame(data = np.array(Lp),
        index = self.data.index[self.startIndex:
        self.finalIndex - self.v])
        return self.LpTS

    def persistenceEntropyTimeSeries(self,d=1):
        PETS = []
        for pc in self.pointClouds:
            try:
                diagram = ripser(pc)['dgms']
                if d == 0:
                    diagram = diagram[d][:-1]
                else:
                    diagram = diagram[d]
                PETS.append(pe.persistent_entropy(diagram, normalize=True))
            except:
                PETS.append(0)
        self.PETS = pd.DataFrame(data = np.array(PETS),index =
        self.data.index[self.startIndex:self.finalIndex - self.v])
        return self.PETS


    def VaRTimeSeries(self, bssize = 100):
        bs_vars = np.array([[np.percentile(
        np.random.choice(self.data.LogRet[i:self.v+i],l
        en(self.data.LogRet[i:self.v+i])), 5, interpolation="lower")
        for j in range(0,bssize)] for i in
        range(self.startIndex,self.finalIndex - self.v)])
        VaR = pd.DataFrame(data = np.array([np.mean(s_var)
        for s_var in bs_vars]),
        index = self.data.index
        [self.startIndex:self.finalIndex - self.v])
        return VaR

    def ESTimeSeries(self, bssize = 100):
        bs_ret = np.array([[np.random.choice(self.data.LogRet[i:self.v+i],
        len(self.data.LogRet[i:self.v+i])) for j in range(0,bssize)]
        for i in range(self.startIndex,self.finalIndex - self.v)])
        bs_vars = np.array([np.percentile(bs, 5, interpolation = 'lower')
        for bs in bs_ret])
        worst_ret = np.array([bs_ret[i][bs_ret[i] <= bs_vars[i]]
        for i in range(len(bs_ret))])
        ES = np.array([np.mean(wr) for wr in worst_ret])
        return pd.DataFrame(data = ES, index = self.data.index
        [self.startIndex:self.finalIndex - self.v])

    def integrateSplines(self, f, x_sample):

        t = [0,1]
        c = 1
        s = 0

        resolution = len(x_sample)

        for i in range(2,resolution):
            if c == 1 and f[i] < f[t[1]]:
                c = -1
                s += (x_sample[t[1]] - x_sample[t[0]])*(f[t[1]] + f[t[0]])
                t = [t[1],i]
            elif c == -1 and f[i] >= f[t[1]]:
                c = 1*(f[i] > f[t[1]])
                s += (x_sample[t[1]] - x_sample[t[0]])*(f[t[1]] + f[t[0]])
                t = [t[1],i]
            elif c == 0 and f[i] > f[t[1]]:
                c = 1
                t = [t[1],i]
            else:
                t[1] = i

        return  0.5*(s + (x_sample[t[1]] - x_sample[t[0]])*(f[t[1]] + f[t[0]]))


    def L1(self,landscape,sample_range,kinConsideration = 5, resolution = 1000):
        return sum([self.integrateSplines
        (landscape[resolution*i:resolution*(i+1)], np.linspace(sample_range[0],sample_range[
        for i in range(0,kinConsideration)])

    def Lp(self, landscape, p, sample_range, kinConsideration = 5, resolution = 1000):
        return sum([mt.lp_norm(skfda.FDataGrid
        ([landscape[resolution*i:resolution*(i+1)]],

        np.linspace(sample_range[0],sample_range[1],1000)),p)
        for i in range(0,kinConsideration)])


    def plotIndex(self, col ='b'):
        fig = plt.figure(figsize=(5,2.5))
        plt.plot(self.getClosePrice(),color = col,linewidth=0.5)
        fig.suptitle(self.filename[:-4], fontsize=15)
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('Index Value', fontsize=10)
        plt.grid(True)
        fig.savefig(self.filename[:-4]+'.jpg')


    def plotlogRet(self,col='b'):
        fig = plt.figure(figsize=(5,2.5))
        plt.plot(self.getLogRet(),color = col, linewidth=0.5)
        fig.suptitle(self.filename[:-4], fontsize=15)
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('Logaritmic Returns', fontsize=10)
        plt.grid(True)
        fig.savefig(self.filename[:-4]+'.jpg')

    def plotlandscapesLpNormTimeSeries(self, p = 1, col = 'r'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + ' L' + str(p) + ' Norm', fontsize=15)
        n = self.landscapesLpNormTimeSeries()
        plt.plot(n,label = 'L'+ str(p), linewidth=0.5, color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('Norm Value', fontsize=10)

    def plotSTDTimeSeries(self, col='y'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + ' STDs', fontsize=15)
        plt.plot(self.getSTDTimeSeries(),label = 'STDs', linewidth=0.5, color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('STDs Value', fontsize=10)

    def plotACFTimeSeries(self,col='g'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + ' ACF', fontsize=15)
        plt.plot(self.getACFTimeSeries(),label = 'ACF', linewidth=0.5,
        color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('ACF Value', fontsize=10)

    def plotKurtosisTimeSeries(self,col='g'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + '␣Kurt', fontsize=15)
        plt.plot(self.getKurtosisTimeSeries(),label = 'Kurt', linewidth=0.5,
        color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('Kurt␣Value', fontsize=10)

    def plotpersistenceEntropyTimeSeries(self,col='r'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + '␣Persitence␣Entropy', fontsize=15)
        plt.plot(self.getpersistenceEntropyTimeSeries(),
        label = 'PEntr', linewidth=0.5, color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('PEntr␣Value', fontsize=10)

    def plotVaRTimeSeries(self, col='y'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + '␣VaR␣Bootstrap', fontsize=15)
        plt.plot(self.getVaRTimeSeries(),label = 'VaR', linewidth=0.5, color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('VaR␣Value', fontsize=10)

    def plotESTimeSeries(self, col='o'):
        fig = plt.figure(figsize=(5,2.5))
        fig.suptitle(self.filename[:-4] + '␣ES␣Bootstrap', fontsize=15)
        plt.plot(self.getESTimeSeries(),label = 'ES', linewidth=0.5, color = col)
        plt.grid(True)
        plt.legend()
        plt.xlabel('Date', fontsize=10)
        plt.ylabel('VaR␣Value', fontsize=10)
```

A.2 Plots

Spearman coefficient between std and L1 Norms for $v = 21 : \rho = 0.473$
Spearman coefficient between std and L1 Norms for $v = 63 : \rho = 0.806$
Spearman coefficient between std and L1 Norms for $v = 126 : \rho = 0.866$

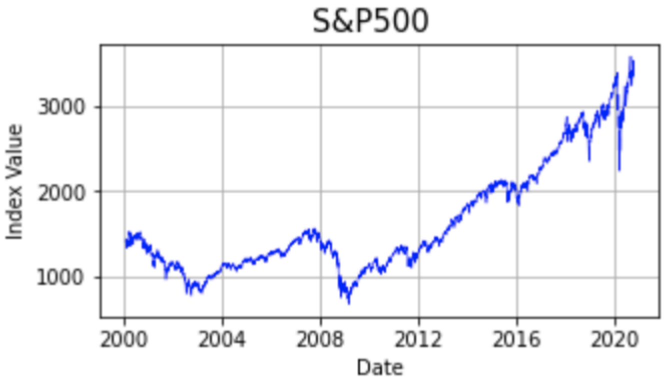


Figure A.1: SP500 index from 01-2000 to 10-2020

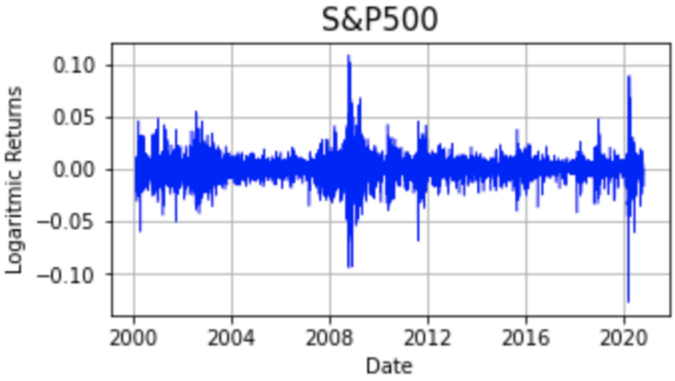


Figure A.2: SP500 log returns from 01-2000 to 10-2020

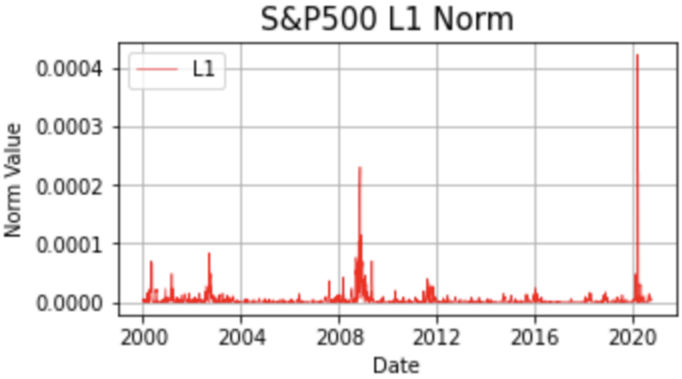


Figure A.3: SP500 L1 Noms, $d = 2$,$v = 21$ from 01-2000 to 10-2020

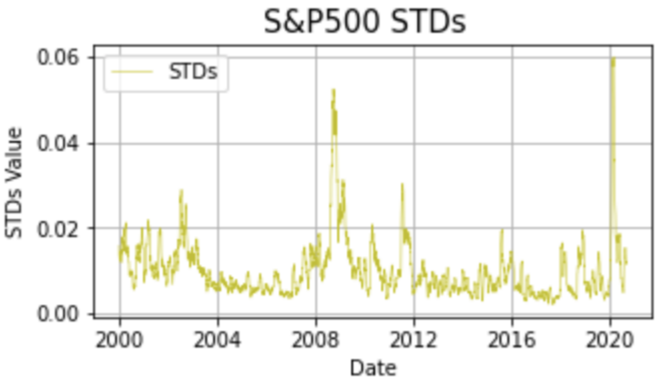


Figure A.4: SP500 L1 Noms, $d = 2$,$v = 21$ from 01-2000 to 10-2020

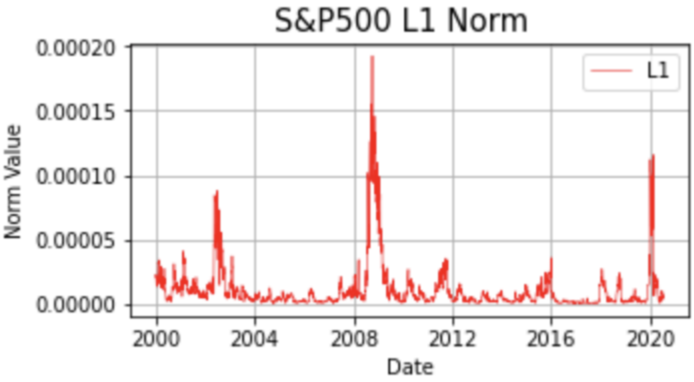


Figure A.5: SP500 L1 Noms, $d = 2$,$v = 63$ from 01-2000 to 10-2020

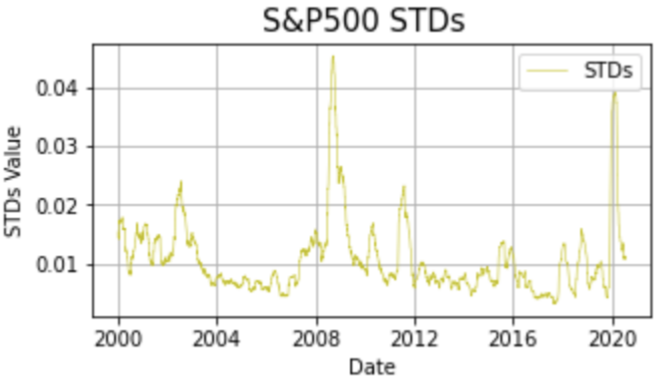


Figure A.6: SP500 L1 Noms, $d = 2$,$v = 63$ from 01-2000 to 10-2020

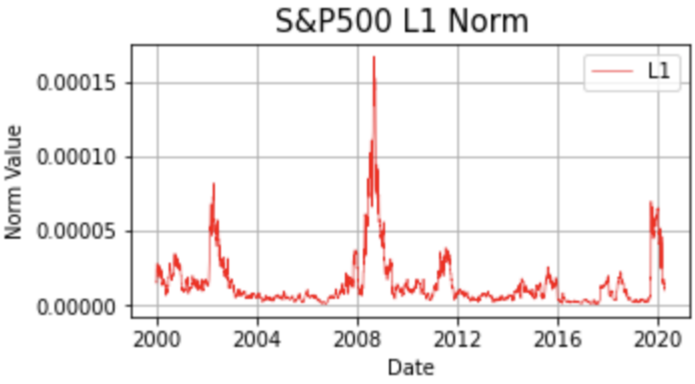


Figure A.7: SP500 L1 Noms, $d = 2, v = 126$ from 01-2000 to 10-2020

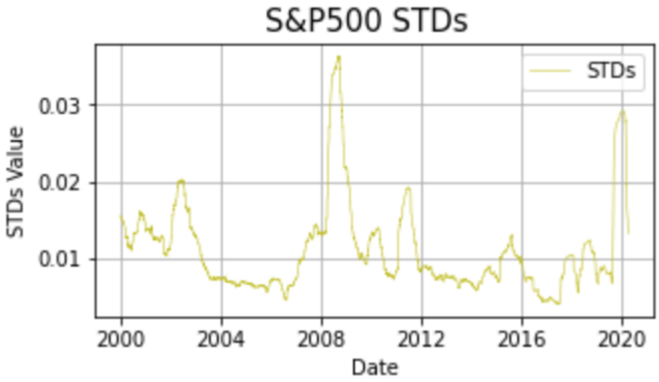


Figure A.8: SP500 L1 Noms, $d = 2, v = 126$ from 01-2000 to 10-2020

Appendix B

TDA Time Series Playground Code

B.1 Code

Listing B.1: Insert code directly in your document

```
import numpy as np
import pandas as pd
import gudhi as gd
import math
import statsmodels.api as sm
import gudhi.representations
from gudhi.point_cloud.timedelay import TimeDelayEmbedding
from gudhi.representations.vector_methods import Landscape
from gudhi.representations.vector_methods import Silhouette
import matplotlib.pyplot as plt
from ripser import ripser
from persim import plot_diagrams
from numpy import linalg as LA
from sklearn import linear_model
import skfda.misc.metrics as mt
import skfda
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import spearmanr,kurtosis,norm
import seaborn as sns
import random
import gtda.mapper as gmp
import persim.persistent_entropy as pe
import statsmodels.tsa.api as smt
from persim import plot_diagrams
import seaborn as sns
```

```
from scipy.stats import kde


class TDATimeSeries():

    def __init__(self,d=2,tau=1):
        self.LS = gd.representations.Landscape(resolution=1000, num_landscapes = 5)
        self.TD = TimeDelayEmbedding(dim = d,delay = tau)

    def plotEmbeddedTimeSeries(self,timeSeries):
        embedded = self.TD.__call__(timeSeries)
        x = [p[0] for p in embedded]
        y = [p[1] for p in embedded]
        fig = plt.figure(figsize=(5,5))
        fig.suptitle('Embedded␣Time␣Series', fontsize=15)
        plt.scatter(x,y,color='b',s=0.2)
        plt.grid(True)

    def plotEmbeddedTimeSeriesDensity(self,timeSeries):
        embedded = self.TD.__call__(timeSeries)
        x = [p[0] for p in embedded]
        y = [p[1] for p in embedded]

        nbins=300
        k = kde.gaussian_kde([x,y])
        xi, yi = np.mgrid[x.min():x.max():nbins*1j, y.min():y.max():nbins*1j]
        zi = k(np.vstack([xi.flatten(), yi.flatten()]))

        plt.pcolormesh(xi, yi, zi.reshape(xi.shape))
        plt.show()

        plt.pcolormesh(xi, yi, zi.reshape(xi.shape), cmap=plt.cm.Greens_r)
        plt.show()

    def persistenceDiagram(self,timeSeries,maxdim=2):
        embedded = self.TD.__call__(timeSeries)
        return ripser(embedded,maxdim)['dgms']

    def plotPersistenceDiagram(self,timeSeries,maxdim=2):
        plot_diagrams(self.persistenceDiagram(timeSeries,maxdim),show=True)

    def plotPersistenceDiagramDensity(self,timeSeries,maxdim=2):
        diagram = self.persistenceDiagram(timeSeries)

        x = np.array([x[0] for x in diagram[1]])
        y = np.array([x[1] for x in diagram[1]])
        nbins=300
        k = kde.gaussian_kde([x,y])
        xi, yi = np.mgrid[x.min():x.max():nbins*1j, y.min():y.max():nbins*1j]
        zi = k(np.vstack([xi.flatten(), yi.flatten()]))

        plt.pcolormesh(xi, yi, zi.reshape(xi.shape))
        plt.show()

        plt.pcolormesh(xi, yi, zi.reshape(xi.shape), cmap=plt.cm.Greens_r)
        plt.show()

    def persistenceLandscapes(self,timeSeries,p=1):
        embedded = self.TD.__call__(timeSeries)
        try:
            diagram = self.persistenceDiagram(timeSeries)
            landscape = self.LS.fit_transform([diagram[1]])[0]
        except:
            landscape = None
        return landscape

    def persistenceLandscapesLpNorm(self,timeSeries,p=1,
    landscapesInConsideration = 5, samplePointsNumber = 1000):

        landscape = self.persistenceLandscapes(timeSeries,p)
        if landscape is None:
            return 0
        sample_range = self.LS.sample_range
        return sum([mt.lp_norm(skfda.FDataGrid(
        [landscape[samplePointsNumber*i:samplePointsNumber*(i+1)]],
        np.linspace(sample_range[0],sample_range[1]
        ,samplePointsNumber)),p) for i in range(0,landscapesInConsideration)])


    def powerWeightedSilhouette(self,timeSeries,p=1):
        embedded = self.TD.__call__(timeSeries)
        acX = gd.AlphaComplex(points=embedded).create_simplex_tree()
        dgmX = acX.persistence()
        SH = gd.representations.Silhouette(resolution=1000, weight=lambda x: 1)
        return SH.fit_transform([acX.persistence_intervals_in_dimension(1)])[0]


    def powerWeightedSilhouetteLpNorm(self,timeSeries,Lp=1,p=1):
        silhouette = silhouette(timeSeries,p)
        return mt.lp_norm(skfda.FDataGrid(
        silhouette[0],
        np.linspace(silhouette.sample_range[0],silhouette.sample_range[1],1000)),p
        )
    def persistenceEntropy(self,timeSeries,d):
        embedded = self.TD.__call__(timeSeries)
        diagram = self.persistenceDiagram(timeSeries,maxdim=1)[d]
        return pe.persistent_entropy(diagram,normalize=True)


    def ES(self, timeSeries,d):

        diagram = self.persistenceDiagram(timeSeries)

        if d == 0:
            diagram = diagram[0][:-1]
        else:
            diagram = diagram[1]

        l = [d[1] - d[0] for d in diagram]
        L = sum(l)
        xo = sorted(diagram, key = lambda x: x[0])[0][0]
        xf = sorted(diagram, key = lambda x: x[1])[-1][1]
        instants = np.linspace(xo,xf,1000)
        diagrams  = [[d for d in diagram if d[0] <= t and t <= d[1]]
        for t in instants]
        dls = [[i[1] - i[0] for i in di] for di in diagrams]
        ES = np.array([sum([-(l/L)*math.log(l/L) for l in ls]) for ls in dls])
        return instants, ES

    def NES(self,timeSeries,d):
        instants, ES = self.ES(timeSeries,d)
        NES = ES/(mt.lp_norm(skfda.FDataGrid(ES,instants),1))
        return instants, NES



import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from scipy.stats import semicircular



def ar_1(n_samples, corr, mu=0, sigma=1):
    assert 0 < corr < 1, "Auto-correlation␣must␣be␣between␣0␣and␣1"

    c = mu * (1 - corr)
    sigma_e = np.sqrt((sigma ** 2) * (1 - corr ** 2))
    signal = [c + np.random.normal(0, sigma_e)]

    for _ in range(1, n_samples):
        signal.append(c + corr* signal[-1] + np.random.normal(0, sigma_e))

    return np.array(signal)


def arch_1(n_samples,a0,a1):

    w = np.random.normal(size = n_samples)
    eps = np.zeros_like(w)
    sigsq = np.zeros_like(w)

    for i in range(1, n_samples):
        sigsq[i] = a0 + a1*(eps[i-1]**2)
        eps[i] = w[i] * np.sqrt(sigsq[i])

    return eps

def garch_1(n_samples,a0,a1,b1):

    w = np.random.normal(size = n_samples)
    eps = np.zeros_like(w)
    sigsq = np.zeros_like(w)

    for i in range(1, n_samples):
        sigsq[i] = a0 + a1*(eps[i-1]**2) + b1*sigsq[i-1]
        eps[i] = w[i] * np.sqrt(sigsq[i])

    return eps

def laplaceTS(n_samples,mu=0, variance=1):
    return np.random.laplace(0, math.sqrt(variance/2), n_samples)

def logisticTS(n_samples,mu=0, variance=1):
    return np.random.logistic(mu, math.sqrt(variance*3)/math.pi, n_samples)

def normalTS(n_samples,mu=0, variance=1):
    return np.random.normal(mu, variance, n_samples)

def semicircularTS(n_samples,mu=0, variance=1):
    return semicircular.rvs(loc = mu,scale = 2*math.sqrt(variance),size=n_samples)

def uniformTS(n_samples,variance = 1):
    return np.random.uniform(0,2*math.sqrt(variance*3),n_samples)
```

```
def tsplot(y, lags=None, figsize=(10, 8), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)
    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)

        layout = (3, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))
        qq_ax = plt.subplot2grid(layout, (2, 0))
        pp_ax = plt.subplot2grid(layout, (2, 1))

        y.plot(ax=ts_ax)
        ts_ax.set_title('Time␣Series␣Analysis␣Plots')
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax, alpha=0.5)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax, alpha=0.5)
        sm.qqplot(y, line='s', ax=qq_ax)
        qq_ax.set_title('QQ␣Plot')
        scs.probplot(y, sparams=(y.mean(), y.std()), plot=pp_ax)

        plt.tight_layout()
    return

    def std_corr_persistence_landscape_lp_norm_surface(
    n_samples,n_sim,corr0=0.00001,corr1=0.99999,n_corrs=10,
    std0=1,std1=10,n_std=10):

    tda = TDATimeSeries()
    autocorrelations = np.linspace(corr0,corr1,n_corrs)
    stds = np.linspace(std0,std1,n_std)
    empiricalStds = []
    fixed_autocorr_LpNormSeries = []

    for corr in autocorrelations:
        fixed_std_LpNormSeries = []
        for std in stds:
            timeSeries = ar_1(n_samples,corr,sigma=std)
            fixed_std_LpNormSeries.append(
            np.mean(np.array(
            [tda.persistenceLandscapesLpNorm(timeSeries) for _ in range(0,n_sim)])))
        fixed_autocorr_LpNormSeries.append(fixed_std_LpNormSeries)
    fixed_autocorr_LpNormSeries = np.array(fixed_autocorr_LpNormSeries)

    x = [x for x in autocorrelations for i in range(len(stds))]
    y = [y for i in range(len(autocorrelations)) for y in stds]
    z = fixed_autocorr_LpNormSeries.ravel()
```

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(x,y,z)
```

B.2 Plots

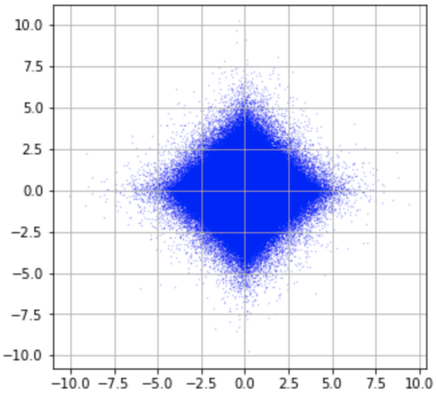


Figure B.1: Laplace i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 3$ for $N = 10000000$.

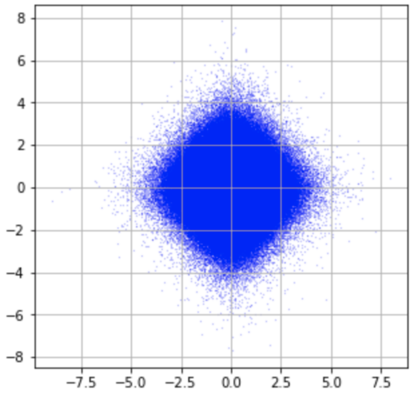


Figure B.2: Logistic i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 1.2$ for $N = 10000000$.

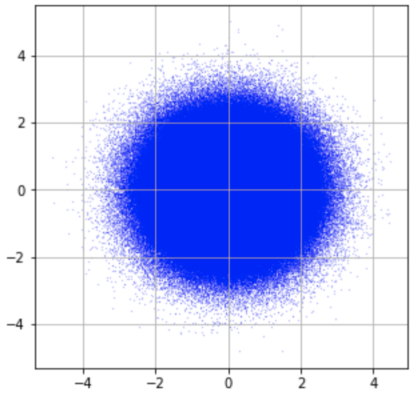


Figure B.3: Normal i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 0$ for $N = 10000000$.

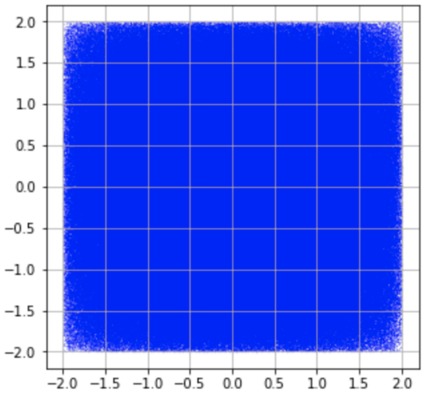


Figure B.4: Semicircular i.i.d noise with $\mu = 0, \sigma = 1, \kappa = 3$ for $N = 10000000$.

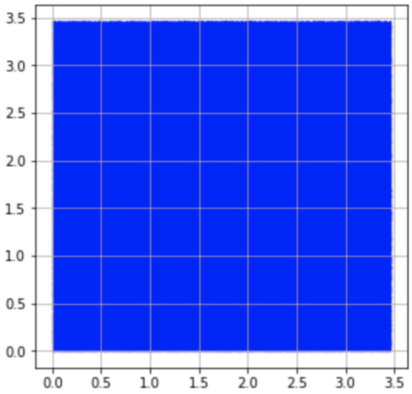


Figure B.5: Uniform i.i.d noise with $\mu = 0, \sigma = 1, \kappa = -1.2$ for $N = 10000000$.

Bibliography

[1] LLoyd Aromi, Yuri Katz, and Josep Vives. "Dependency of functional norms of persistence landscapes on variance-covariance: application to multivariate time series". In: (2020).

[2] Nieves Atienza, Rocio Gonzalez-Díaz, and Manuel Soriano-Trigueros. "On the stability of persistent entropy and new summary functions for topological data analysis". In: *Pattern Recognition* 107 (Nov. 2020), p. 107509. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107509. URL: http://dx.doi.org/10.1016/j.patcog.2020.107509.

[3] Eric Berry et al. "Functional Summaries of Persistence Diagrams". In: *Journal of Applied and Computational Topology* 4 (June 2020). DOI: 10.1007/s41468-020-00048-w.

[4] Leo Betthauser, Peter Bubenik, and Parker B. Edwards. *Graded persistence diagrams and persistence landscapes*. 2020. arXiv: 1904.12807 [math.AT].

[5] Peter Bubenik. "Statistical Topological Data Analysis Using Persistence Landscapes". In: *J. Mach. Learn. Res.* 16.1 (Jan. 2015), pp. 77–102. ISSN: 1532-4435.

[6] Peter Bubenik. "The Persistence Landscape and Some of Its Properties". In: *Abel Symposia* (2020), pp. 97–117. ISSN: 2197-8549. DOI: 10.1007/978-3-030-43408-3_4. URL: http://dx.doi.org/10.1007/978-3-030-43408-3_4.

[7] Peter Bubenik and Pawel Dlotko. "A persistence landscapes toolbox for topological statistics". In: *Journal of Symbolic Computation* 78 (Jan. 2017), 91â114. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2016.03.009. URL: http://dx.doi.org/10.1016/j.jsc.2016.03.009.

[8] Gunnar Carlsson. "Topology and Data". In: *Bulletin of The American Mathematical Society - BULL AMER MATH SOC* 46 (Apr. 2009), pp. 255–308. DOI: 10.1090/S0273-0979-09-01249-X.

[9] David Cohen-Steiner et al. "Lipschitz Functions Have Lp-Stable Persistence". In: *Foundations of Computational Mathematics* 10 (Feb. 2010), pp. 127–139. DOI: 10.1007/s10208-010-9060-6.

[10] Frédéric Chazal et al. "Stochastic Convergence of Persistence Landscapes and Silhouettes". In: *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*. SOCG'14. Kyoto, Japan: Association for Computing Machinery, 2014, 474â483. ISBN: 9781450325943. DOI: 10.1145/2582112.2582128. URL: https://doi.org/10.1145/2582112.2582128.

[11] Harish Chintakunta et al. "An entropy-based persistence barcode". In: *Pattern Recognition* 48 (Feb. 2015). DOI: 10.1016/j.patcog.2014.06.023.

[12] Stefan Dantchev and Ioannis Ivrissimtzis. "Efficient construction of the Čech complex." In: *Computers & graphics.* 36.6 (Mar. 2012), pp. 708–713. URL: http://dro.dur.ac.uk/28442/.

[13] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction.* American Mathematical Society, 2010, pp. I–XII, 1–241. ISBN: 978-0-8218-4925-5.

[14] Edelsbrunner, Letscher, and Zomorodian. "Topological Persistence and Simplification". In: *Discrete & Computational Geometry* 28.4 (2002), pp. 511–533. DOI: 10.1007/s00454-002-2885-2. URL: https://doi.org/10.1007/s00454-002-2885-2.

[15] Bernd Gärtner. "Fast and Robust Smallest Enclosing Balls". In: *Proceedings of the 7th Annual European Symposium on Algorithms*. ESA '99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 325–338. ISBN: 3540662510.

[16] Robert Ghrist. *Barcodes: The persistent topology of data*. Tech. rep. 2007.

[17] Marian Gidea and Yuri Katz. "Topological data analysis of financial time series: Landscapes of crashes". In: *Physica A: Statistical Mechanics and its Applications* 491 (Feb. 2018), 820â834. ISSN: 0378-4371. DOI: 10.1016/j.physa.2017.09.028. URL: http://dx.doi.org/10.1016/j.physa.2017.09.028.

[18] Anubha Goel, Puneet Pasricha, and Aparna Mehra. "Topological data analysis in investment decisions". In: *Expert Systems with Applications* 147 (2020), p. 113222. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113222. URL: http://www.sciencedirect.com/science/article/pii/S0957417420300488.

[19] Jan Grandell. *Time Series Analysis*. URL: http://www.math.kth.se/matstat/gru/5b1545/ts.pdf.

[20] Allen Hatcher. *Algebraic topology*. Cambridge: Cambridge Univ. Press, 2000.

[21] David Leroy Johnson. *Topics in the Theory of Group Presentations*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1980. DOI: 10.1017/CBO9780511629303.

[22] Firas Khasawneh and Elizabeth Munch. “Chatter detection in turning using persistent homology”. In: *Mechanical Systems and Signal Processing* 70 (Oct. 2015). DOI: 10.1016/j.ymssp.2015.09.046.

[23] Firas Khasawneh and Elizabeth Munch. “Chatter detection in turning using persistent homology”. In: *Mechanical Systems and Signal Processing* 70 (Oct. 2015). DOI: 10.1016/j.ymssp.2015.09.046.

[24] Jisu Kim et al. *Homotopy Reconstruction via the Cech Complex and the Vietoris-Rips Complex*. 2020. arXiv: 1903.06955 [math.AT].

[25] Michael Ledoux and Michael Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. A Series of Modern Surveys in Mathematics Series. Springer, 1991. ISBN: 9783540520139.

[26] Tom Leinster. *Basic Category Theory*. 2016. arXiv: 1612.09375 [math.CT].

[27] Yuriy Mileyko, Sayan Mukherjee, and John Harer. “Probability measures on the space of persistence diagrams”. In: *Inverse Problems* 27.12 (Nov. 2011), p. 124007. DOI: 10.1088/0266-5611/27/12/124007. URL: https://doi.org/10.1088/0266-5611/27/12/124007.

[28] Elizabeth Munch et al. “Probabilistic Fréchet means for time varying persistence diagrams”. In: *Electronic Journal of Statistics* 9 (Jan. 2015), pp. 1173–1204. DOI: 10.1214/15-EJS1030.

[29] Raymundo Navarrete. “Embeddings and Prediction of Dynamical Time Series”. In: 2018.

[30] Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. “Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival”. In: *Proceedings of the National Academy of Sciences* 108.17 (2011), pp. 7265–7270. ISSN: 0027-8424. DOI: 10.1073/pnas.1102826108. eprint: https://www.pnas.org/content/108/17/7265.full.pdf. URL: https://www.pnas.org/content/108/17/7265.

[31] Jose Perea and John Harer. “Sliding Windows and Persistence: An Application of Topological Methods to Signal Analysis”. In: *Foundations of Computational Mathematics* 15 (July 2013). DOI: 10.1007/s10208-014-9206-z.

[32] Nalini Ravishanker and Renjie Chen. *An exploration of topological properties of high-frequency one-dimensional financial time series data using TDA*. 2017.

[33] Nalini Ravishanker and Renjie Chen. *Topological Data Analysis (TDA) for Time Series*. 2019. arXiv: 1909.10604 [stat.AP].

[34] Michael Kerber René Corbet. "The Representation Theorem of Persistence Revisited and Generalized". In: *Journal of Applied and Computational Topology (2018)* (). DOI: 10.1007/s41468-018-0015-3.

[35] Rodrigo Rivera-Castro, Polina Pilyugina, and Evgeny Burnaev. "Topological Data Analysis for Portfolio Management of Cryptocurrencies". In: *2019 International Conference on Data Mining Workshops (ICDMW)* (Nov. 2019). DOI: 10.1109/icdmw.2019.00044. URL: http://dx.doi.org/10.1109/ICDMW.2019.00044.

[36] Matteo Rucco et al. "Characterisation of the idiotypic immune network through persistent entropy". In: (Jan. 2015).

[37] Martin Sewell. *Characterization of Financial Time Series*. 2011.

[38] Claude E. Shannon. "A mathematical theory of communication." In: *Bell Syst. Tech. J.* 27.3 (1948), pp. 379–423. URL: http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48.

[39] Vin de Silva and Robert Ghrist. "Coverage in sensor networks via persistent homology". In: *Algebr. Geom. Topol.* 7.1 (2007), pp. 339–358. DOI: 10.2140/agt.2007.7.339. URL: https://doi.org/10.2140/agt.2007.7.339.

[40] Vin Silva and Robert Ghrist. "Coverage in sensor networks via persistent homology". In: *Algebraic Geometric Topology* 7 (Apr. 2007). DOI: 10.2140/agt.2007.7.339.

[41] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. "Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition". In: *Eurographics Symposium on Point-Based Graphics*. Ed. by M. Botsch et al. The Eurographics Association, 2007. ISBN: 978-3-905673-51-7. DOI: 10.2312/SPBG/SPBG07/091-100.

[42] John Stillwell. *Classical Topology and Combinatorial Group Theory*. Graduate Texts in Mathematics. Springer New York, 2012. ISBN: 9781468401103. URL: https://books.google.es/books?id=oFTxBwAAQBAJ.

[43] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by David Rand and Lai-Sang Young. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 366–381. ISBN: 978-3-540-38945-3.

[44] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. "Ripser.py: A Lean Persistent Homology Library for Python". In: *The Journal of Open Source Software* 3.29 (Sept. 2018), p. 925. DOI: 10.21105/joss.00925. URL: https://doi.org/10.21105/joss.00925.

[45] Josep Vives. *Lecture Notes on Time Series Analysis*. 2020.

[46] Afra Zomorodian. "Fast construction of the Vietoris-Rips complex". In: *Computers Graphics* 34.3 (2010). Shape Modelling International (SMI) Conference 2010, pp. 263–271. ISSN: 0097-8493. DOI: `https://doi.org/10.1016/j.cag.2010.03.007`. URL: `http://www.sciencedirect.com/science/article/pii/S0097849310000464`.

[47] Afra Zomorodian and Gunnar. Carlsson. "Computing Persistent Homology". In: *Discrete & Computational Geometry* (2005). DOI: `10.1007/s00454-004-1146-y`.

[48] Afra J. Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005. DOI: `10.1017/CBO9780511546945`.