Analyzing financial time series with persistent homology

JOHANNA OTT and EDOUARD VILAIN

ACM Reference Format:
Johanna Ott and Edouard Vilain. 2022. Analyzing financial time series with persistent homology . 1, 1 (April 2022), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROJECT SCOPE

The goal of this project is to analyze the evolution of daily returns of four major US stock markets indices (DowJones, Nasdaq, Russell2000, SP500) over the period 1989 − 2016 using persistent homology, following the approach proposed in [1]. A classical approach in TDA to extract topological features from multivariate time-series with values in Rd (d = 4 here, since we are considering the evolution of four indices) consists in using a sliding window of fixed length w to generate a sequence of w points in Rd. Using the Vietoris-Rips filtration, the persistence diagram of each of these point clouds is then computed and used as a topological feature for further analysis or processing of the initial data. This project aims at reproducing the experiments of [1] and explore and discuss a few variants.

2 PERSISTENCE LANDSCAPES

As indicated in the abstract and introduction, the persistence diagrams built with Rips filtration are the heart of this analysis. In this section, we are going to give an introduction into this topic and describe how we have implemented it.

Introduction

In this section, we are going to restate the definitions of Rips, filtration, homology, persistence diagrams and eventually persistence landscapes according to [1] and XXX ... cite lecture notes. Given a point cloud X in R^d, a Vietoris-Rips or simply Rips complex is defined as the topological space R(X,ϵ) containing the points of X which also build the vertices of this space and with the distance ϵ>0 if the following holds true:

$$d(x_i, x_j) < \epsilon; \text{ for all } x_i, x_j \text{ in X}$$

the distance between any points of X is within ϵ distance.
Furthermore, a filtration of a complex is a nested sequence of subcomplexes starting at the empty set and ending with the full simplicial complex. For a For the Rips simplicial complexes R(X,ϵ) this means that R(X,ϵ) $\subseteq$ R(X,ϵ') if ϵ<ϵ'. In the following graphic we visualize such a Rips filtration. We see that the four given points belong to different Rips complex while the ϵ increases from left to right.

Based on these Rips complex we can compute the corresponding k-dimensional homology, $H_k(R(X, \epsilon))$. They benefit from the same filtration properties: $H_k(R(X, \epsilon)) \subseteq H_k(R(X, \epsilon'))$ if ϵ<ϵ'.

Authors' address: Johanna Ott; Edouard Vilain.

XXXX-XXXX/2022/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

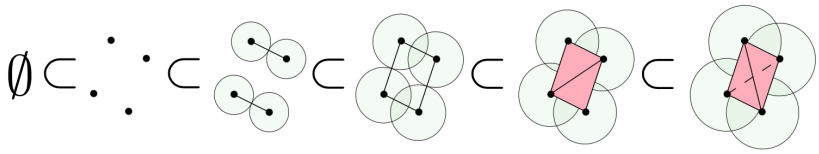

Fig. 1. Rips Filtration Example [2]

We can then apply Rips Filtration to all k-dimensional homologies and encode them into persistence diagrams P_k. A persistence diagrams summarizes topological information. For each k-dimensional homology class α, we can associate a pair $(b_\alpha, d_\alpha) \in \mathbb{R}^2$ of respectively birth and death indices. Essentially, their existence is ensured using the filtration property mentioned above and they correspond respectively to the smallest and largest value of ϵ for which $\alpha \in H_k(R(X,\epsilon))$. The induced persistence diagram P_k is thus a set of pairs $P_k = \left\{(b_\alpha, d_\alpha) | \alpha \in \bigcup_{\epsilon \geq 0} H_k(R(X,\epsilon))\right\}$ which can be represented in the plane.

Though the space of persistence diagrams P endowed with the Wasserstein distance $\mathcal{W}_p$ forms a metric space $(P, \mathcal{W}_p)$, it is not complete thus isn't an adapted space for statistical treatment. This is the reason why we introduce persistence landscapes which enable a representation of persistence diagrams in the Banach space $L^p(\mathbb{N} \times \mathbb{R})$.

Let $(b_\alpha, d_\alpha) \in P_k$, we define the following function:

$$f_{(b_\alpha, d_\alpha)}(x) = \begin{cases} x - b_\alpha & \text{if } x \in [b_\alpha, \frac{b_\alpha + d_\alpha}{2}] \\ -x + d_\alpha & \text{if } x \in]\frac{b_\alpha + d_\alpha}{2}, d_\alpha] \\ 0 & \text{if } x \notin [b_\alpha, d_\alpha] \end{cases}$$

Fig. 2. Persistence diagram to Landscape example [1]

Consider a persistence diagram P_k with a finite number of off-diagonal points. Let $l \in \mathbb{N}$, we define the function $\lambda_l : \mathbb{R}[0,1]$ as the following:

$$\lambda_l(x) = l - max\left\{f_{(b_\alpha, d_\alpha)}(x) | (b_\alpha, d_\alpha) \in P_k\right\}$$

where $l - max$ defines the l-th largest value of a set. For any $l > |P_k|$, we set $\lambda_l = 0$. Finally, the persistence landscape λ is defined as the family $(\lambda_l)_{l \in \mathbb{N}}$ of $L^p(\mathbb{N} \times \mathbb{R}), \forall p \in \mathbb{N}^*$. When endowed with the L^p norm, the space of persistence landscapes lies within a Banach space.

Implementation of Persistence Landscapes

In this section we discuss the paradigm choices for the implemented function with which the persistence landscapes get calculated. For that purpose we wrote a function ourself and afterwards verified it through a comparison with the GUDHI-based version.

Requirements.
The persistence landscape is supposed to be implemented with the following parameters: an persistence diagram D in the GUDHI format, a dimension k, the endpoints x_{min}, x_{max} of an interval, the number n of nodes of a regular grid on the interval $[x_{min}, x_{max}]$ and a number of landscapes m.

The persistence diagram in GUDHI is represented as a list of tuples, with each tuple corresponding to a homology class. The latter is represented as follows: $(dimension, (b_\alpha, d_\alpha))$ where $dimension$ is the dimension of the homology class α and (b_α, d_α) are respectively the birth and death values of α. The dimension k represents the dimension of the homology classes we consider. As mentionned above, a persistence diagram P_k is computed with regards to the homologies of dimension k, thus, the persistence landscape also.

Implementation without GUDHI.
We now discuss our implementation choice. In the spirit of the Gudhi framework, we decide to implement the persitence diagram in an Object Oriented manner. We thus define a class $PersistenceLandscape$ with the following characteristics:

(1) **Attribute** k (integer): the homology dimension.
(2) **Attribute** x_{min}, x_{max} (float): the endpoints of the interval.
(3) **Attribute** n (integer): the number of nodes in the regular grid.
(4) **Attribute** m (integer): the number of landscapes.
(5) **Attribute** $landscapes$ (integer array): an array containing the values of the m landscapes at each of the n nodes of the regular grid.
(6) **Method** $build\_landscape$: takes as input the parameters mentionned above, computes the landscapes matrix and updates the attributes of the class.

Note that, because of the current simplicity of the class, taking an Object Oriented approach is not necessary. We still decided to follow this approach considering it offers the best framework to further complexity and add methods.
We now describe the $build\_landscape$ algorithm used to build the persistence landscape of a given persistence diagram.It takes as an input D persistence diagram, k dimension of the homology, x_{min}, x_{max} the endpoints of the interval, n the number of nodes in the regular grid and m the number of landscapes. The output is an updated $PersistenceLandscape$ object. The algorithm contains the following steps:

(1) Filter through D to keep homology classes of dimension k (we name the resulting list $D_{filtered}$).
(2) Compute $f_{(b_\alpha, d_\alpha)}$ on the regular grid defined by x_{min}, x_{max}, n for each homology class α in $D_{filtered}$. This induces a matrix f of dimension $|D_{filtered}| \times n$.
(3) Order each column of f in descending order. Each row of the resulting $f_{ordered}$ corresponds to the $l - max$ vector of f over the grid. Thus, we have computed the $|D_{filtered}|$ first persistence landscape vectors of our point cloud.
(4) Compute the persistence landscapes matrix $landscapes$ by restricting or expanding the number of rows of $f_{filtered}$ to m.
(5) Update the attributes of the $PersistenceLandscape$ object.

Naturally, we will need to implement an annex function $compute\_f$ which takes a couple of birth and death values b_α, d_α, a real value x and returns $f_{(b_\alpha, d_\alpha)}(x)$.

Verification of the algorithm.
We tested our algorithm against simple examples and the GUDHI-based Implementation. First, we sampled a normally distributed cloud of points in a plane. We then use GUDHI to apply Rips

filtration and obtain the resulting persistence diagram for dimensions 0 and 1. Finally, we compute the persistence landscapes using the GUDHI method and our implemented solution. We compare the results of both solutions to ensure the correctness of our approach. This comparison can be seen in 3.

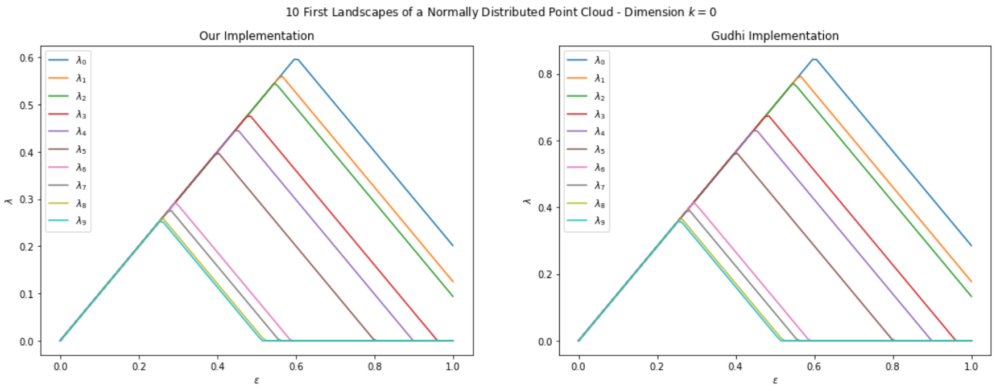

Fig. 3. Comparison of Landscapes Implementations

We notices a slight difference in the plots and investigated it by plotting the difference between the two landscapes which can be seen in 4.

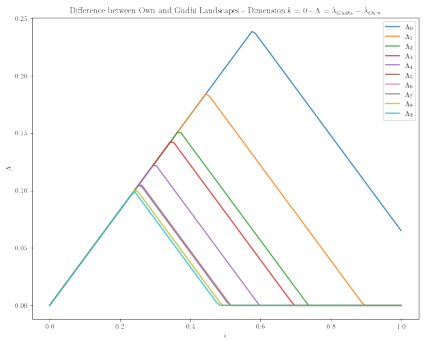

Fig. 4. Difference in Implementations

It appeared that there is a linear difference between our and GUDHI's implementations regarding the slopes of the landscapes. After looking at the source code of GUDHI's Landscape class, we were able to verify furthermore a clear difference in how values of $f_{(b_\alpha,d_\alpha)}$ are computed over the regular grid.

While we define a function that computes $f_{(b_\alpha,d_\alpha)}$ explicitly for each value of x over the regular grid, GUDHI first computes the indexes of the three rupture points of $f_{(b_\alpha,d_\alpha)}$ on the regular grid. It then computes the values of $f_{(b_\alpha,d_\alpha)}$ sequentially by incrementing then decrementing by a constant step $\eta = \frac{x_{max}-x_{min}}{resolution}$. At first sight, the latter method seems more prone to errors as it makes more approximations while computing the rupture indexes. Indeed, b_α, d_α, $\frac{b_\alpha+d_\alpha}{2}$, in general, do not align with the regular grid. Moreover, GUDHI also defines the increment and decrements values using

these approximations. This offset can be translated by a linearly increasing difference between the theoretical values of $f_{(b_\alpha, d_\alpha)}$ over the grid and the computed ones.

A simple way to compare the correctness of both algorithms is thus to study the slopes, birth and death values of each computed landscapes. Visually, it seems like both algorithms coincide on the birth and death values of the landscapes. We will therefore concentrate our efforts on the landscapes' slopes. Considering $f_{(b_\alpha, d_\alpha)}$ have a slope of 1 or -1 when non-vanishing, and, by definition of the landscapes λ_k, computed over a finite set of off-diagonal points, the non-vanishing parts of each λ_k should also have a slope of 1 or -1. We thus plot both landscapes against a linear slope of equation $y = x$ and compare the observations we received in 5.

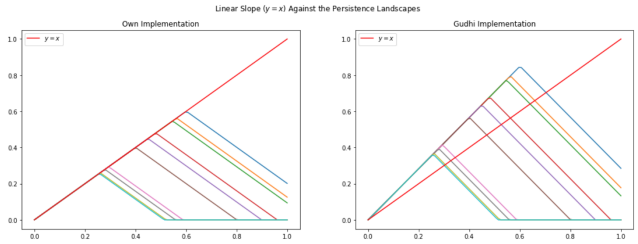

Fig. 5. Difference in Slopes

We observe that our landscapes are much better aligned with the linear equation and all have an increasing slope of 1. Given the observations made above, we can ensure the correctness of our approach and the lack of precision in GUDHI's implementation.
Moreover, the lack of precision in GUDHI's approach does not appear to give any benefit complexity wise. Indeed, in both algorithms, values of $f_{(b_\alpha, d_\alpha)}$ are computed sequentially and in a linear manner. Therefore, it seems as our implementation gives better precision without compromise in computational time.

3 ANALYZING THE EVOLUTION OF DAILY RETURNS OF MAJOR US STOCK MARKET

In this section, we aim to reproduce the experiments led in Section [1] which analyze the daily time series of four US stock markets (SP 500, DJIA, NASDAQ, Russell 2000) between December 23rd 1987 and December 8th 2016. They aim to apply TDA (Topological Data Analysis) and specifically study the properties of persistence homologies around the major economical crisis of the dotcom crash on 03/10/2000 and the Lehman bankruptcy on 09/15/2008. If successful, this work could introduce a new type of economic analysis able to detect imminent market crashes. We will follow the same steps as in the article using our implementation of the persistence landscapes. We will also continuously compare our results to the ones obtained by the article.

Data Overview

We first summarize the article's approach to the matter at hand. The study focuses on the daily log returns $r_{ij} = \log\left(\frac{P_{i,j}}{P_{i-1,j}}\right)$ of the adjusted closing values $P_{i,j}$ of each index j and day i. This induces a multivariate time series of $\mathbb{R}^4$ for which we wish to explore persistence homologies properties. To do so, we introduce a sliding window w (the article considers $w = 50$ and 100 and we will consider $w = 40, 80$ and 120). We then slide the window accross the time series and consider the persistence properties of each of the sub-series induced by the sliding window. Each of these sub-series describes as a point cloud of $\mathbb{R}^4$ of size w which can be represented in a $w \times 4$ size matrix.

We represent the point cloud induced by indexes DJIA and NASDAQ around different days and different window sizes in 6 and 7.

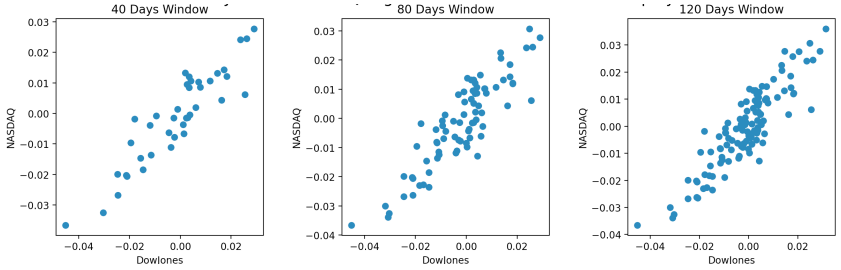

Fig. 6. DowJones vs NASDAQ Log-Returns Before Lehman Crisis

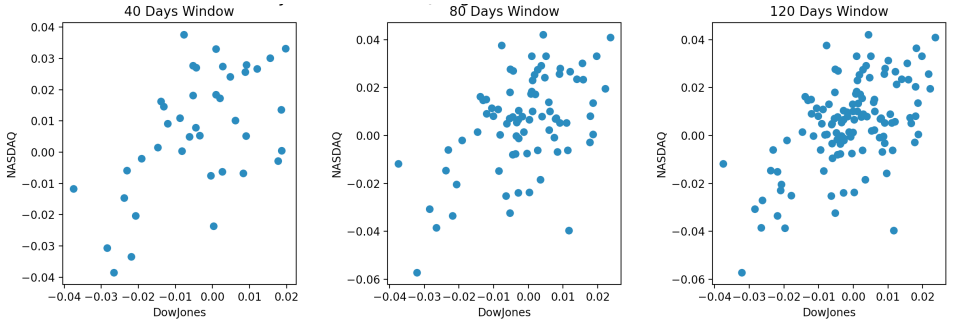

Fig. 7. DowJones vs NASDAQ Log-Returns Before Dotcom Crash

Data to L-norms

For each of the sub-series defined by the sliding windows, we compute the Rips filtration of the induced point cloud along with the persistence diagram and persistence landscapes. We first apply Rips filtration to all windows and compute their persistence diagrams. They then get visualized as can be seen in 8 .
The red dots in 8 correspond to connected components while blue dots correspond to loops. We now enlarge the scope by computing all persistence diagrams for all window sizes. Since this is a computational intensive process, we tracked the computational times. We observed that time for building persistence diagrams increases exponentially with regards to the size of a point cloud. Thus, window sizes cannot be chosen too large because of feasibility issues.

From these persistence diagrams, we can compute the induced persistence landscapes. We do that both using our method and GUDHI's method and compare the performance in time. We decide upon the following lanscape parameters:

(1) k (dimension of homology classes): we will compute persistence landscapes both for connected component ($k = 0$) and loop ($k = 1$) homology classes.
(2) x_{min}, x_{max} (interval endpoints): all log-return values reside within a sphere (in dimension 4) of diameter 0.24. We thus decide to use $x_{min} = 0$ and $x_{max} = 0.24$.

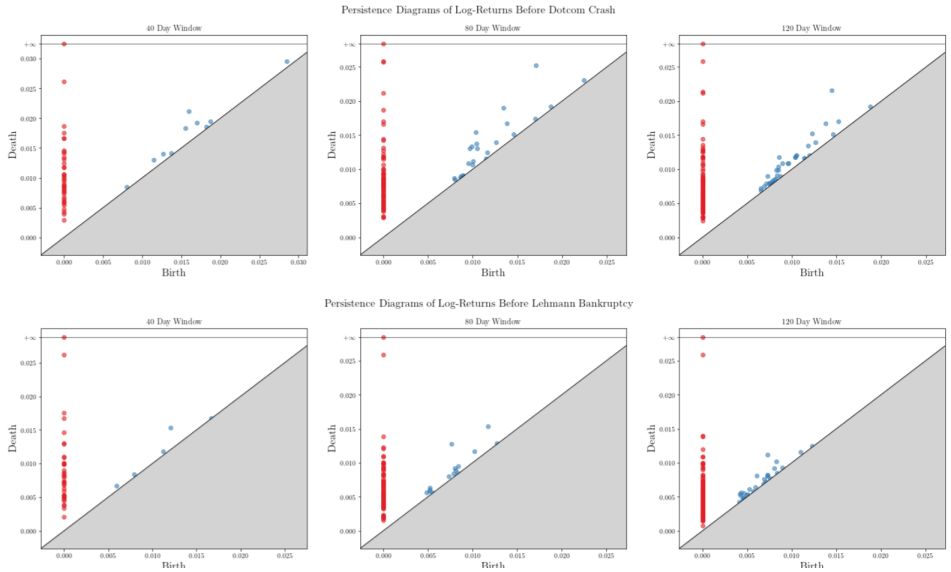

Fig. 8. Persistence Diagrams of Log-Returns Before Lehmann Bankruptcy

(3) m (number of landscapes): for a good compromise between precision and computation time, we decide to use $m = 15$ for all window sizes.
(4) n (resolution of the regular grid): given the number of landscapes, a resolution of $n = 50$ seems reasonable.

Not only did we improve the correctness of Gudhi's persistence landscape computation, but we were also able to improve Gudhi's computation time. On average, our computations were 2.25 times faster than Gudhi. Moreover, the time gained increases with the size of the point cloud. We have all the reasons to believe our approach to implementing Persistence Landscapes has more advantages than Gudhi's.
We visualized the persistence landscapes for each sub-time series and for each window size. For that we focused on the time windows preceding the two major crisis mentioned in the article. The result is visualized in the abstract in 12. From these figures arise two main observations. First, persistence landscapes for homologies of dimension 1 vanish extremely fast. This is also the case for most other windows which generate a low amount of 1-dimensional homology classes. Second, it also seems like persistence landscapes tend to vanish faster with an increasing window size. This is probably caused by the high density of the point cloud induced by such a window. We will further study the influence of the window size in the following experiments. To compare with the article, the rest of our study will focus on the homology classes of dimension 1. The authors do not specify the reasons that led them to this choice. Most likely, it was motivated by the information loops bring to a cloud of points rather than holes.
By computing the L^1 and L^2 norms of the inferred persistence landscapes, we will compute the daily time series of these measurements. We hope these new signals will help us in identifying specific properties of financial series before market crashes. First, we remind the definition of the

L^p norm. Let $p > 0$, $\lambda = (\lambda_l)_{l \in \mathbb{N}}$ be persistence landscapes of a given point cloud, then:

$$\|\lambda\|_p = \left(\sum_{l=1} \|_l\|_p^p \right)^{\frac{1}{p}}$$

We compute the induced time series. Similarly to the study led by the article, we observe that our L^1 and L^2 signals differ significantly. Indeed, both signals seem to produce the same up- and down-trends, especially with increasing window sizes. Moreover, high values of w produce smoother signals while smaller values produce signals with high noise and many vanishing values. The latter appear for small window sizes because of the lack of loop structures in smaller cloud points. As the size of a cloud point increases (and consequently the window size) appear more loop homology classes and less vanishing persistence landscapes.
We now visualize the signals in the 3 years preceding the Dotcom Crash (03/10/2000) and the Lehman Bankruptcy (09/15/2008) in 9. We analyze the behaviour of the L^1 and L^2 norms before these major financial crisis.

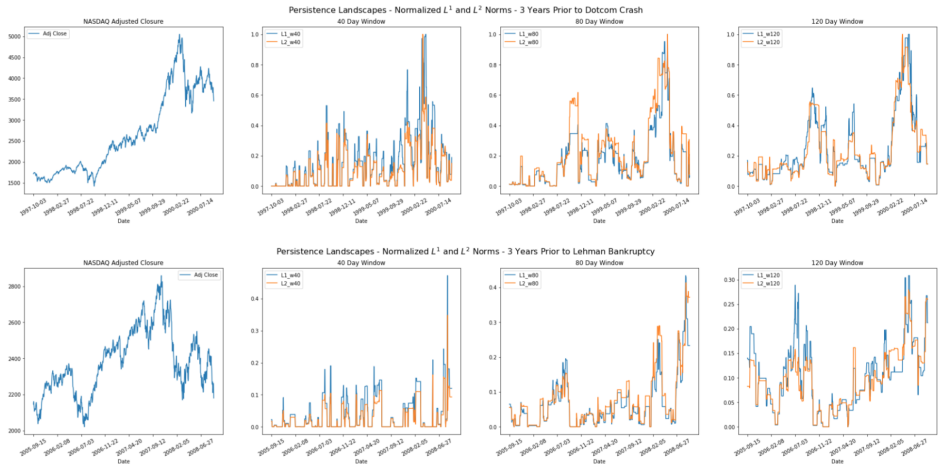

Fig. 9. Persistence Landscapes - Normalized three years before the crashes

Similarly to the observations made in the article, we observe that both L^1 and L^2 norms demonstrate strong fore-shocks prior to the primary peaks. While the article only mentions L^1 signals, we observe the same behaviour with the L^2 norm. In fact, while L^1 norms have very abrupt peaks, the ones from the L^2 signals are squarer and may be easier to identify. Moreover, we make the same observations as above regarding window sizes. With an increasing window size, signals become smoother thus up- and down-trends become much more identifiable. In the same time, less fore-shocks appear but the most singnificant ones also become easier to identify. Therefore, with higher window sizes, we are able to preserve the most relevant information while remove most of the irrelevant one. This suggests a compromise must be found between precision and computational complexity of high window sizes. Finally, we do not observe any evident shift of the prior maximum towards the later dates.

4 RESULTS

Like in the article, we applied after all these pre-processing steps the standard technique and analyze the leading statistical indicators of the L^p norms of persistence landscapes. Inspired from [1], we

compute the variance, the average spectral densities at low frequencies of our signal and the first lag of the Auto-Correlation Function (ACF). All of these information are computed over a period of 250 trading days prior to the dates of the two main market crashes. Note that no indication is given regarding the range that is considered for low-frequencies of the spectral densities (neither in the main article or [1]). We thus decide upon a range of the first 10% of frequencies. The following plots,11 and 10, capture these information.

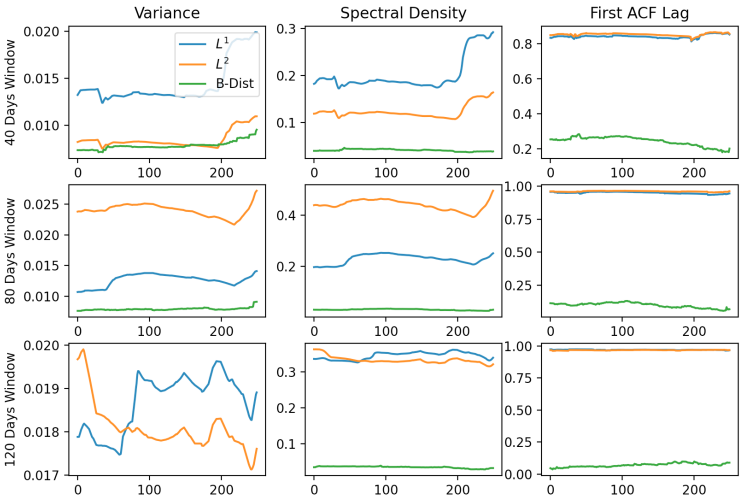

Fig. 10. Statistical Indicators of L^1 and L^2 Norms Bottleneck Distance (B-Dist) - 250 Days Prior to Dotcom Crash

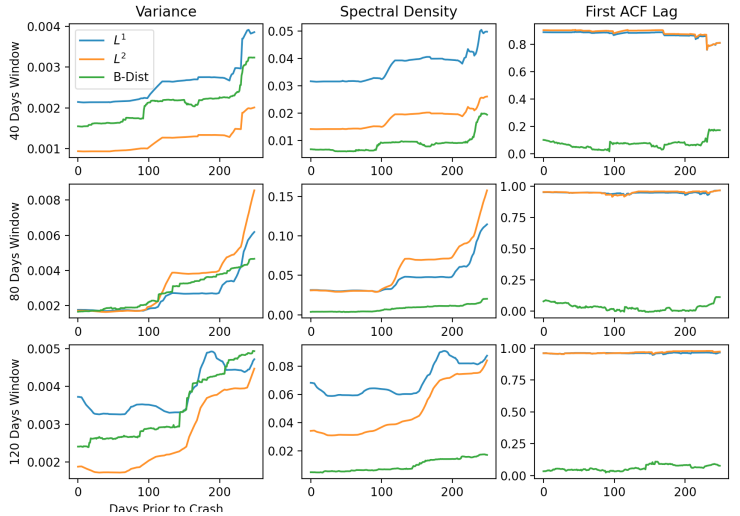

Fig. 11. Statistical Indicators of L^1 and L^2 Norms and Bottleneck Distance (B-Dist) - 250 Days Prior to Lehman Bankruptcy

Our results are extremely similar to the ones produced by the article. We are able to identify the same characteristics of variance and spectral density in the L^1 and L^2 norm curves.

We experimented further and calculated the statistical measures also on the basis of the bottleneck distance between consecutive daily returns of the respective persistence diagrams. The results are also visualized in 11 and 10. The "Bottleneck distance measures the similarity between two persistence diagrams. It's the shortest distance b for which there exists a perfect matching between the points of the two diagrams (+ all the diagonal points) such that any couple of matched points are at distance at most b, where the distance between points is the sup norm in $\mathbf{R}^2$."[3] Similar to the norms we also see detect patterns, uptrends, indicating the crashes.

Final Remarks and Comparison to [1]

First, similarly to the article, we do not detect any characteristic behaviour regarding first lags of the ACFs prior to both crashes. In fact, their evolution over time seems to be quite noisy and irregular.
Second, like in the article, we are able to detect uptrends in the variance and spectral density properties of the L^1 and L^2 norm series. These uptrends are extremely abrupt and, in both cases, begin around 50 days prior to the crash. It is reasonnable to suppose these uptrends become detectable at least 10 days prior to the crash which represents a significant advantage in financial terms.
Finally, the experiments led above suggest there is no correct window size, but rather that several window sizes could prove complementary in such an analysis. Indeed, though uptrends are identifiable in both crashes using windows of 40 days, we observe that higher window sizes in the case of the Lehman Bankruptcy make the uptrends much more identifiable. Indeed, in the case of the 80 days window, the uptrends in variance and spectral density are much more abrupt and identifiable than for the smaller window. Moreover, while using a 120 day window, we observe an extremely neat increase in variance and spectral density for the L^2 norm. This trend initiates 100 days prior to the crash and is clearly identifiable at least 75 days prior to the crash.
In conclusion, our observations suggest that working with several window sizes offer a great opportunity to identify major economic crises. Indeed, while the article considers using windows of 50 and 100 days produces the same results, we have shown that, by using three windows of sizes 40, 80 and 120, both the Dotcom Crash and the Lehman Bankruptcy would have been more predictable. This means that not only uptrends in the statisical properties of L^1 and L^2 norms were clearer to identify, but they would also have been identified much earlier.

5 ABSTRACT

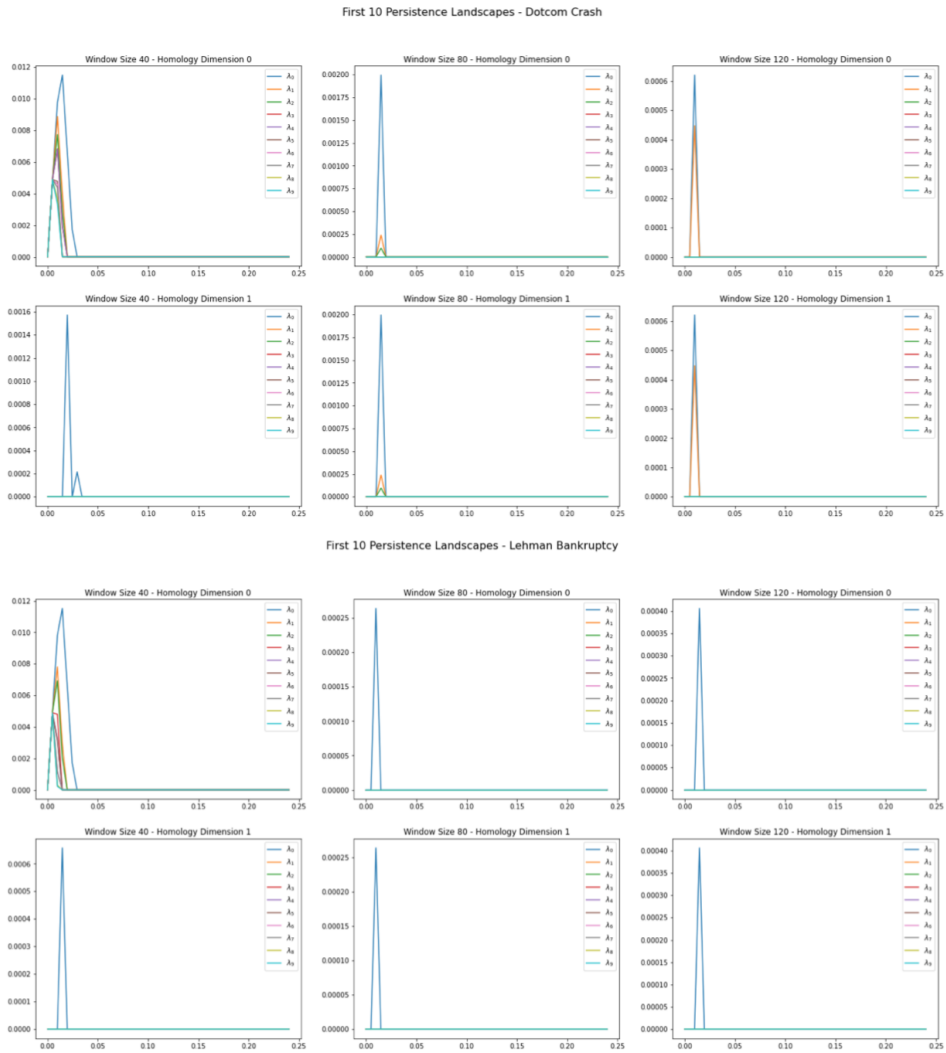

Fig. 12. Persistence Landscapes before Crashes

REFERENCES

[1] Marian Gidea and Yuri Katz. 2018. Topological data analysis of financial time series: Landscapes of crashes. *Physica A: Statistical Mechanics and its Applications* 491 (feb 2018), 820–834. https://doi.org/10.1016/j.physa.2017.09.028

[2] Ben Holmgren. 2019. persistent_homology. https://comptag.github.io/rpackage_tutorials//2019/04/persistent_homology.html Accessed: 2022-04-15.

[3] The GUDHI Project. 2015. *GUDHI User and Reference Manual.* GUDHI Editorial Board. http://gudhi.gforge.inria.fr/doc/latest/