DRAFT VERSION APRIL 14, 2020
Typeset using LaTeX **twocolumn** style in AASTeX63

Extracting the main trend in a dataset

Dalya Baron[1] and Brice Ménard[2]

[1]*School of Physics and Astronomy, Tel-Aviv University, Tel Aviv 69978, Israel*
[2]*Department of Physics and Astronomy, The Johns Hopkins University, 3400 N Charles Street, Baltimore, MD 21218, USA*

(Dated: April 14, 2020)

Abstract

Scientists aim to extract simplicity from observations of the complex world. An important component of this process is the exploration of data in search of trends. In practice, however, this tends to be more of an art than a science. Among all trends existing in the natural world, one-dimensional trends, often called sequences, are of particular interest as they provide insights into simple phenomena. However, some are challenging to detect as they may be expressed in complex manners. We present the Sequencer, an algorithm designed to generically identify the main trend in a dataset. It does so by constructing graphs describing the similarities between pairs of observations, computed with a set of metrics and scales. Using the fact that continuous trends lead to more elongated graphs, the algorithm can identify which aspects of the data are relevant in establishing a global sequence. Such an approach can be used beyond the proposed algorithm and can optimize the parameters of any dimensionality reduction technique. We demonstrate the power of the Sequencer using real-world data from astronomy, geology as well as images from the natural world. We show that, in a number of cases, it outperforms the popular t-SNE and UMAP dimensionality reduction techniques. This approach to exploratory data analysis, which does not rely on training nor tuning any parameter, has the potential to enable discoveries in a wide range of scientific domains.

Keywords: data analysis | statistical | graph

1. Introduction

"One of the principal objects of theoretical research is to find the point of view from which the subject appears in the greatest simplicity", wrote Josiah Willard Gibbs in 1881. The early phase of this process often involves exploratory data analysis, i.e. a search for patterns in a dataset without the benefit of guidance from theory. Unfortunately, data can appear complex and might not allow underlying trends to be revealed straightforwardly. Additional challenges include high dimensionality, the presence of noise, and ever-growing data volumes, all of which prevent efficient visualization of the data and require mathematically guided exploration.

Dimensionality reduction techniques, from Principal Component Analysis (PCA; Pearson 1901) to the more recent t-Distributed Stochastic Neighbor Embedding (t-SNE; van der Maaten & Hinton 2008) and Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP; McInnes et al. 2018), provide powerful ways to address some of these limitations (Van Der Maaten et al. 2009; Lee & Verleysen 2010; Venna et al. 2010). However, scientific work does not end once dimensionality reduction algorithms have been applied to a dataset. Rather, it only begins. Extracting simplicity through the identification of interesting trends critically relies on user input and judgement. Domain knowledge informs important decisions: the choice of coordinates, the scale to focus on, the metric to use to compare objects, etc. In addition, when the size or dimensionality of data objects is large, such is the case for images, spectra or time series, analyses are typically performed on extracted features or summary statistics rather than on measured values or pixels. This decision is critical in the exploration process since a poor choice of summary statistics or relevant features might prevent the detection of interesting trends in the data. Finally, most dimensionality reduction algorithms depend on parameters, and changing them can dramatically affect the resulting representation (Wattenberg et al. 2016; McInnes et al. 2018; Baron 2019). Thus, a crucial part of the scientific exploration process is devoted to understanding which observables or summary statistics to investigate, which method to pick, and which parameter values to use, in order to obtain

dalyabaron@gmail.com, menard@jhu.edu

the most interesting results. In many cases, we lack a well-defined metric to guide these decisions (e.g., Lee & Verleysen 2010; Zhang et al. 2011; Baron 2019). Without theoretical guidance, trial and error is usually the adopted strategy. Is it possible to perform these tasks in a way that will automatically and robustly identify the existence of a simple trend in an apparently complex dataset? Is it possible to operate directly on the raw data, without specifying which observables or summary statistics to use?

When trying to understand a dataset, one attempts to extract meaning by building a generic and concise representation of it. For the representation to be generic, it should be invariant to coordinates transformation and deformations. Meaningful trends are topological properties of the data, i.e. aspects of the data manifold that are not affected by deformation (Carlsson 2009). For example, clusters can often be defined and interpreted, irrespective of the choice of coordinates. Many cluster finding algorithms are available and widely used (e.g., Ward 1963; MacQueen 1967; Yizong Cheng 1995; Ester et al. 1996; Rodriguez & Laio 2014). However, in many cases, we expect observed phenomena to exhibit a continuous change in their properties as a function of a leading, possibly unknown, driving parameter. In other words, we often expect to find sequences – they abound in the natural world and, especially, in scientific measurements. Although they are essentially one-dimensional trends, they are often challenging to find as they can be expressed in complex manners.

In this paper we present an algorithm to automatically detect sequences in datasets. It uses information about the shape of the graph describing the similarities between the objects and exploits the fact that sequences give rise to elongated graphs. Following this approach, it is possible to consider different representations of the data and, for each of them, quantify the degree to which a continuous trend exists. We show that this method can also be used to optimize some of the parameters or certain arbitrary choices involved in dimensionality reduction techniques aimed at detecting sequences. Importantly, this search can be performed directly on the pixels or measured values, as opposed to user-defined observables or restrictive summary statistics. Therefore, our approach enables a more generic search for continuous trends in arbitrary datasets.

2. The signature of a continuous trend

Data acquisition often provides us with a collection of objects that are not necessarily ordered in a meaningful manner. If these objects follow an underlying trend due to the variation of an intrinsic parameter, it should be possible to order the set meaningfully. If the variation of this parameter leads to a continuous change of observables, the ordered set should minimize the cumulative differences between consecutive objects. Finding the ordering leading to such minimization is therefore expected to reveal the leading trend in a dataset. By doing so, we face several challenges: which aspects of the data should be considered? Which pixels carry relevant information? How to meaningfully define similarities between them?

To address these questions, we propose the following approach: let us consider a collection of N objects and let us first assume that we have a useful metric allowing us to estimate the similarity (or equivalently a distance) between each pair of objects. The corresponding adjacency matrix represents a fully-connected graph, where each object in the original dataset is represented by a node in the graph, and the weights of the edges that connect the nodes represent the distances between the objects. This structure encodes all the possible trajectories within the dataset.

Within this set, some trajectories are of special interest: those that connect all the nodes and minimize the total distance accumulated along them[1]. We can find such a trajectory by finding the minimum spanning tree (MST) of our graph, the subset of the edges in a fully-connected graph that connects all the nodes together, without any cycles, and with the minimum possible *total* edge weight (e.g., Kruskal 1956). If each edge in the original fully-connected graph has a distinct weight, then the minimum spanning tree is unique. The shape of the resulting tree carries valuable information on the topological properties of the dataset. In particular, it can be used as an indicator of the degree to which a sequence exists in the dataset. In Figure 1 we show examples of minimum spanning trees for three scenarios. The left panel shows the minimum spanning tree of a random graph. The middle panel shows the minimum spanning tree of a dataset with a noisy sequence, and the right panel shows the minimum spanning tree of a dataset with a perfect sequence. The existence of a continuous trend leads to a more elongated minimum spanning tree. Therefore, *the minimum spanning tree elongation can characterize the existence of a trend* underlying a collection of objects.

When doing exploratory data analysis, one typically does not know a-priori how to meaningfully "look" at the data. Which similarity measure will be informative? On which scales will we find relevant information? Interestingly, we can address these questions by simply considering, in each case, the shape of the corresponding minimum spanning tree. In other words, we can automatically find the parameters that are most sensitive to the existence of a simple trend in the dataset. For a diverse enough set of distance metrics

[1] This is not equivalent to solving the Traveling Salesman's Problem. The Traveling Salesman's Problem is more specific and aims at finding a trajectory which starts and ends at the same node and visits all the others only once.

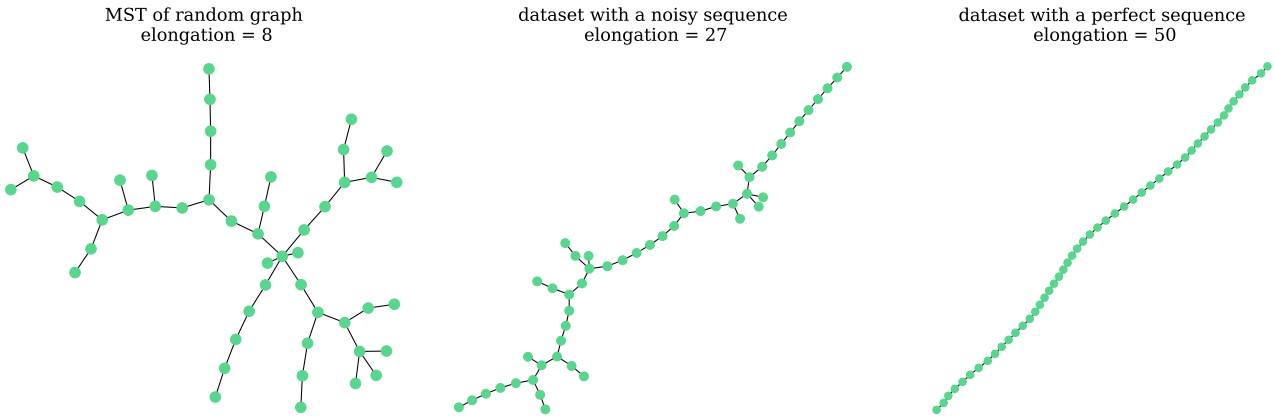

Figure 1. Minimum spanning trees for three sets of 50 objects for which the similarities range from random (left) to continuous (right). Each panel shows the corresponding elongation parameter η (Eq. 3) which can be used to characterize the degree to which a continuous trend is detected.

and scales, this automatic process has the capacity to reveal trends in a generic manner. These trends can be intrinsic to the objects in the sample and/or extrinsic and driven by observational effects.

3. Algorithm description

We now provide a description of the Sequencer algorithm following the key principles outlined in the previous section. Our goal is to order a collection of $N_{\rm obj}$ objects with $N_{\rm pix}$ values. We will describe each object as X_i^j, with $j = 1$ to $N_{\rm obj}$ and $i = 1$ to $N_{\rm pix}$. To characterize and extract a sequence underlying such a collection of objects, we proceed in two steps:

1. For a list of metrics and scales, we compute the corresponding graphs describing the dataset and quantify the elongation of their minimum spanning trees;
2. We aggregate the results to form a new graph summarizing the relevant information and extract an ordered list of objects from it.

As we will demonstrate, this can reveal the trend characterizing the leading variation among the objects in many cases. Such a trend is often meaningful.

3.1. Distance metrics & scales

As described above, we can achieve a meaningful ordering of the dataset by minimizing the similarity or distance between adjacent objects. Doing so requires the choice of a distance measure. In order to be generic, we include several commonly-used metrics. This will allow the algorithm to consider various aspects of the data and its features. Similarly, we consider a list of scales on which the relevant information can be distributed.

By default, we use the following metrics: (i) the Euclidean Distance, (ii) the Kullback-Leibler Divergence (KL Divergence; Kullback & Leibler 1951), (iii) the Monge-Wasserstein or Earth Mover Distance (EMD; Rubner et al. 1998), and (iv) the Energy Distance (Székely 2002). The definitions and properties of these metrics are described in the Appendix. If desired by the user, this list can be expanded to better suit a particular application. We note that this default set includes metrics with different properties: the Earth Mover Distance and Energy Distance are sensitive to the magnitude of displacements along the i coordinate. This is important with continuous measurements, for example performed as a function of space or time, for which the derivative with respect to i carries relevant information. In contrast, the Euclidian Distance and the KL Divergence treat the different pixels of X_i as different dimensions and are insensitive to index shuffling. They provide a qualitatively different view of the information content.

The observable signature of an underlying trend can exist on different scales which may not be known a-priori. In order to be generic, it is important to consider a range of scales. To do so, we decompose each object X into a series of contiguous segments whose length is given by $N_{\rm pix}/2^l$. This results in an ensemble of segments which allows us to look at each data object hierarchically, starting from its entirety ($l = 0$) and creating a binary tree such that the deepest scale corresponds to about twenty pixels. Thus, for a given metric k and scale l, the object is split into 2^l segments, and we refer to each segment using the index m. The maximum depth or the ways in which the data is decomposed can be modified by the user if need be.

3.2. Finding the main sequence

Our goal is to look at the data for each metric k, scale l and segment m, and estimate the level to which an underlying trend is present. To do so, we proceed as follows. First, for each scale l and each segment m, we first normalize each

Algorithm 1: Sequencer pseudo-code

```
set list of metrics;
set list of scales;
for each metric k do
    for each scale l do
        set list of segments;
        split objects X into m segments X_m;
        normalize each segment to have a sum of 1;
        for each segment m do
            D_klm = distance matrix(X_m);
            MST_klm = Minimum Spanning Tree(D_klm);
            η_klm = a_klm/b_klm = elongation(MST_klm);
        end
        D_kl = η-weighted average of individual D_klm;
        MST_kl = Minimum Spanning Tree(D_kl);
        η_kl = elongation(MST_kl);
    end
end
P = combined proximity matrix populated by η-weighted
  edges of MST_kl;
for each pair of objects i, j do
    D_ij^combined = 1/P_ij^combined;
end
MST = Minimum Spanning Tree(D);
Sequence = Breadth First Search path(MST);
```

object such that the sum over its components is one. We then extract geometrical properties of the set of graphs characterizing similarities between all pairs of objects:

• **Graph minimum spanning tree:** for each metric k, scale l and segment m, we compute a $N_{\rm obj}^2$ distance matrix D_{klm} which represents a fully-connected graph. We then compute its minimum spanning tree using Kruskal's algorithm (Kruskal 1956). It gives us a set of $k \times l \times m$ trees with $N_{\rm obj}$ nodes. The key information on the presence of an underlying trend resides in the shape of these graphs.

• **Graph length:** The least connected node, $j_{\rm LC}$, of the minimum spanning tree is expected to belong to its longest branch. To identify it, we compute the closeness centrality of each node (Freeman & Freeman 1978) and select the one with the smallest value. We then compute the shortest path $\Delta_{klm}(j_{\rm LC}, j)$ between this node and every other node in the minimum spanning tree. The shortest path is a unitless integer counting the minimal number of edges between the two nodes (see the Appendix for additional details). We then define the major axis of the minimum spanning tree to be the average of the shortest paths over all nodes:

$$a_{klm} = \langle \Delta_{klm}(j_{\rm LC}, j) \rangle_{\text{node } j} \ . \tag{1}$$

• **Graph width:** Each node in the graph can be assigned to a level q which corresponds to a unique value of $\Delta_{klm}(j_{\rm LC}, j)$. That is, the shortest path between all the nodes that are assigned to the level q and the least connected node: $\Delta_{klm}(j_{\rm LC}, j) = \Delta_q$ (see section 7.2 for an illustration). The width of a level q, $\Delta^{\perp}_{klm}(q)$, is defined as the number of nodes that are assigned to it. We use the average of this quantity as an estimate of the average half width, or minor axis b, of the minimum spanning tree:

$$b_{klm} = \frac{1}{2} \langle \Delta^{\perp}_{klm}(q) \rangle_{\text{level } q} \tag{2}$$

• **Graph elongation:** for each metric k, scale l, and segment m, we then define the elongation of the minimum spanning tree as its average height divided by its average width:

$$\eta_{klm} = \text{elongation}(D_{klm}) = \frac{a_{klm}}{b_{klm}} \, . \tag{3}$$

This quantity can then be used to characterize the level to which the signature of a continuous trend is apparent in a given segment of the objects, through a given metric and on a given scale. Importantly, by being a ratio of numbers of edges, this parameter can be defined irrespective of the metric used. It is a summary statistics describing geometrical properties of the minimum spanning tree and, as a result, topological properties of the data.

• **Aggregation of scales and metrics:** each minimum spanning tree represents a sequence viewed through a given metric k and scale l of a segment of the data m. The elongation η_{klm} of its corresponding minimum spanning tree carries information on the level at which an underlying trend is detected. We can first combine the information obtained for all segments by creating a global distance matrix D_{kl} using an elongation-weighted average of our set of minimum spanning trees:

$$D_{kl} = \langle \eta_{klm} \, . \, D_{klm} \rangle_m \ . \tag{4}$$

This provides us with $k \times l$ different "views" of the data, which we can attempt to aggregate. Here we need to keep in mind that different metrics k will result in distance matrices D_{kl} with different units. To meaningfully combine information obtained from different metrics, we will only extract the topological information of the resulting minimum spanning trees, as given by their edge counts, and use an elongation-weighted average to create a "proximity" matrix:

$$P^{\rm combined} = \langle \eta_{kl} \, . \, \#\text{ of edges}(\text{MST}(D_{kl})) \rangle_{kl} \ . \tag{5}$$

We set all the elements which are not populated by the edges of the minimum spanning trees to zero (no connection between the corresponding nodes), and the elements on the diagonal to infinity (the proximity of a node to itself is infinite). This then allows us to define a combined distance matrix whose elements i, j are defined as $D_{ij}^{\rm combined} = 1/P_{ij}^{\rm combined}$. This distance matrix $D^{\rm combined}$ provides us with a multi-scale and multi-metric characterization of the dataset. We then compute its minimum spanning tree and corresponding elongation.

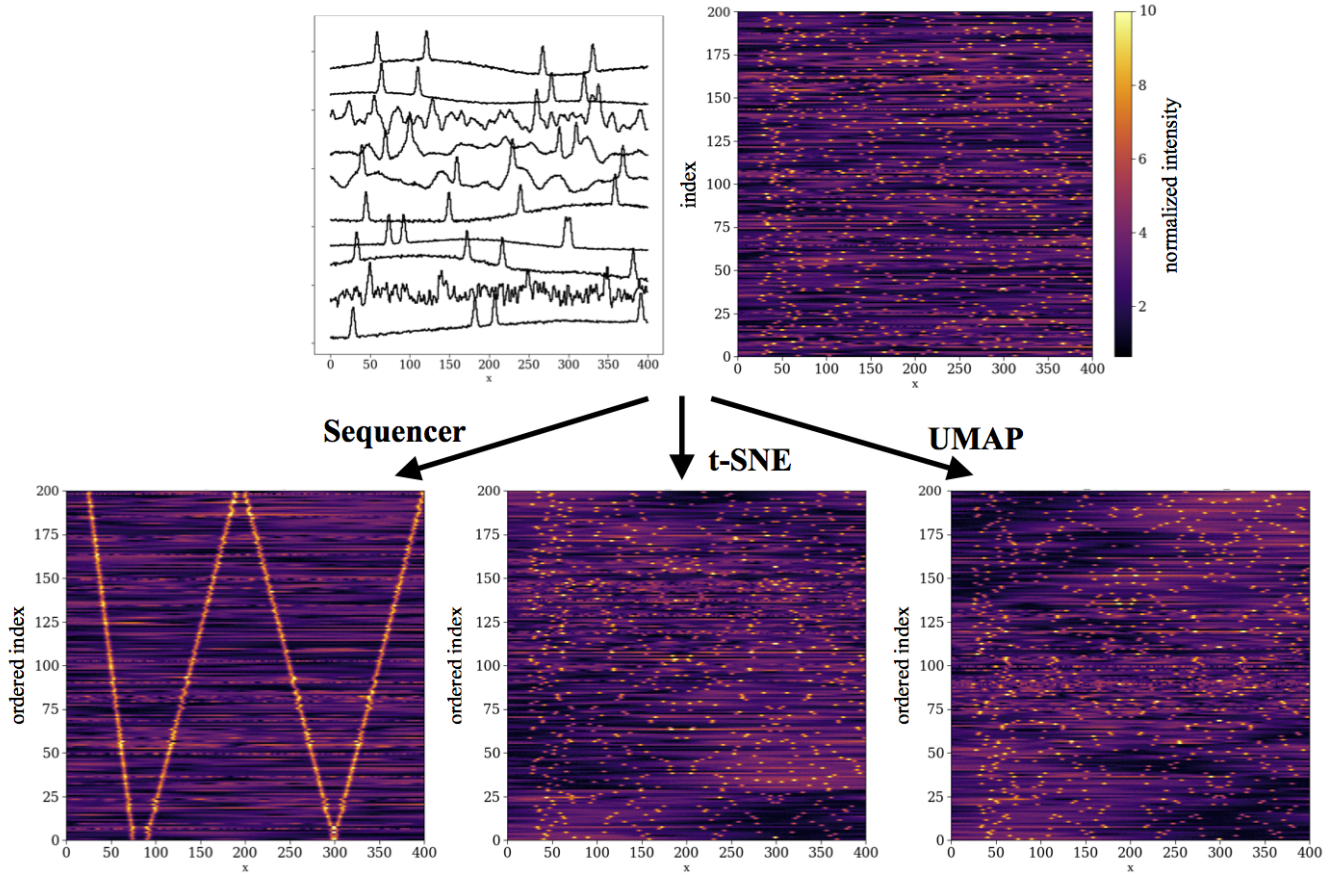

Figure 2. Application of the Sequencer, t-SNE, and UMAP to a simulated dataset with a clear one-dimensional sequence. The top left panel shows examples of objects randomly selected from the simulated dataset. The top right panel shows all the objects from the dataset, where each row represents a single object and is color-coded according to the intensity in each of its pixels. The bottom panels show the objects, ordered according to the Sequencer, t-SNE, and UMAP respectively. It illustrates the ability of the Sequencer to focus on the scale of interest.

• **Extraction of a final sequence:** in order to extract a sequence from the minimum spanning tree of the combined distance matrix, we must select a particular walk within the tree, i.e. we must select the relative order in which we visit all the nodes within the graph. As done above, we define the starting point of the sequence using the least connected node of the combined minimum spanning tree. From this starting point, we walk through the graph using a Breadth First Search (BFS; Cormen et al. 2009). If the combined minimum spanning tree presents an appreciable elongation, a BFS traversal is expected to define the main trend in the dataset. Thus, we expect the main branch of the tree to represent the sequence, and secondary branches to represent the scatter or the secondary sequence.

We point out that the addition of one or several new objects to the dataset can be done without redoing the full process. Once a sequence has been obtained, one can easily insert a new object into it, by performing a straightforward neighbor search. More details are provided in the Appendix.

3.3. Scaling considerations

The algorithm described above requires distance matrix calculations which scale as $\mathcal{O}(N_{\rm obj}^2)$. For large datasets this approach can be computationally demanding. A useful alternative is to apply the technique to a subset of the data with $N_s \ll N$ objects, build the skeleton of a sequence and then populate it with the remaining points. This process leads to a faster computation which allows one to process significantly larger datasets, at the cost of obtaining only an approximate result. A more detailed description of this method is provided in the Appendix.

4. Performance & results

We now apply the algorithm to datasets of increasing complexity to demonstrate its effectiveness as well as its advantages compared to existing dimensionality reduction techniques. In this paper, for simplicity, we will only consider datasets with one dimensional objects. However, the key organizing principle based on the minimum spanning tree elon-

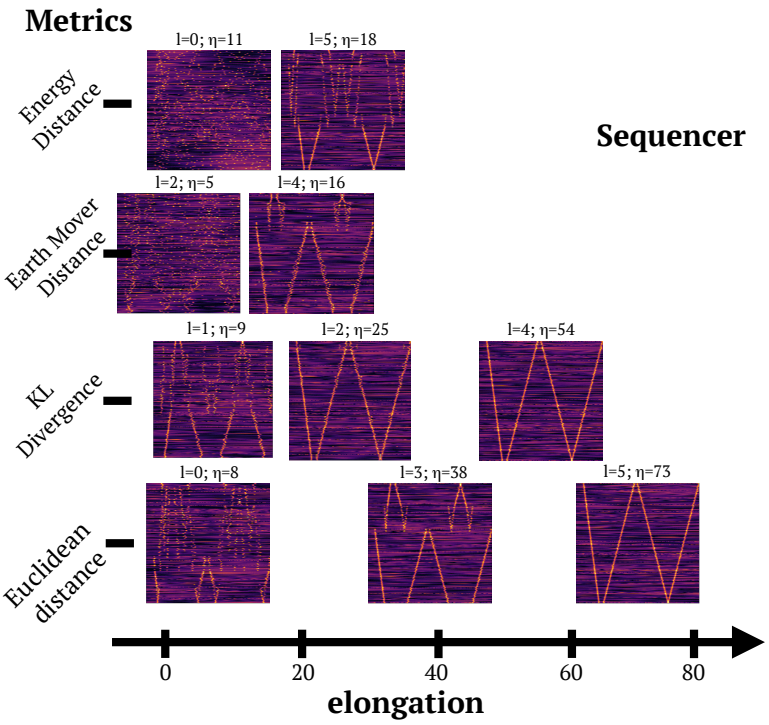

Figure 3. Using the elongation parameter to identify the scale and metric that reveal a meaningful trend in a dataset. The synthetic dataset from figure 2 ordered according to individual distance metrics and scales. The scales and resulting elongation parameters are indicated at the top of each panel.

gation can be applied to objects in two or higher dimensions[2], but at a higher computational cost. Unless specified otherwise, we apply the Sequencer using its default setting, i.e. with the four distance metrics and the binary scale decomposition described in section 3.

4.1. Validation with a simulated dataset

We first construct a synthetic dataset with a well-defined trend that exists only on small scales, on top of a varying background. We create 200 one-dimensional objects with $N_{\rm pix} = 400$ presenting four narrow Gaussian pulses whose positions vary continuously from one object to another to form a clear sequence. To this we add random large-scale fluctuations using a Gaussian process. We show the shuffled dataset in the top panels of Figure 2, where the left panel shows a subset of the objects in the sample, and the right panel visualizes the full dataset. Each row represents a different object, and the color-coding represents the relative intensity in each of its pixels. As can be seen, it is visually difficult to identify an underlying trend in this collection of objects. We apply the Sequencer to this dataset and show its output in the bottom left panel. The algorithm is capable of detecting the overall structure in the dataset even though only a small fraction of the pixels carries relevant information.

For comparison, we show how t-SNE and UMAP, two of the most popular dimensionality reduction techniques, perform on the same simulated dataset. To obtain a sequence using t-SNE or UMAP, we apply these techniques to embed the input dataset into one dimension, and then rank-order the objects according to their assigned value in this dimension. We point out that, as mentioned in the introduction, using these algorithms requires setting parameters. In practice, the user typically runs such an algorithm multiple times, varying those parameters until the "best" result is obtained. Here we present only the best sequences obtained after optimizing the distance metric in an automatic manner, using the elongation of the corresponding graphs. The details of this optimization are given in the Appendix. Clearly, t-SNE and UMAP fail to detect the sequence in narrow pulse locations as they only consider an object as a whole, without the ability to focus on individual segments and understand that a well-defined trend exists on small scales.

To illustrate how the Sequencer identifies the scale and metric revealing a meaningful trend, in Figure 3 we present the resulting ordering of the algorithm for each metric and a set of scales as a function of the corresponding elongation parameter. The most elongated minimum spanning trees are

[2] The publicly-available code can be applied to two-dimensional datasets.

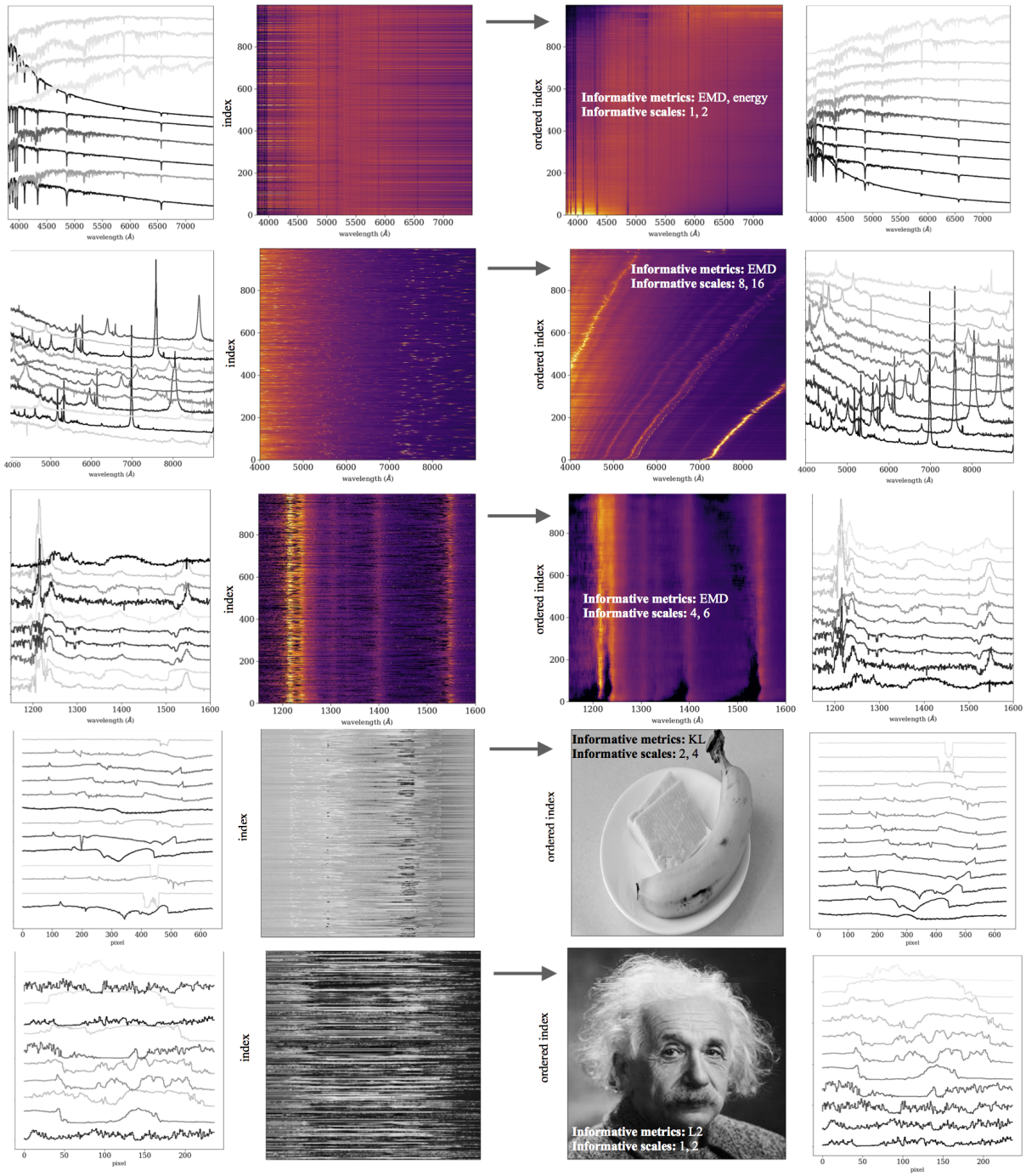

Figure 4. Application of the Sequencer to five randomly-ordered datasets: (a) one thousand spectra of stars, which are then ordered by temperature, (b) one thousand spectra of quasars, which are then ordered by redshift, (c) one thousand spectra of quasars with complex broad absorption line systems, which are then ordered by absorber type and velocity distribution, (d) a picture of food with the rows shuffled, which is then properly reconstructed, and (e) a picture of Albert Einstein with the rows shuffled, which is then properly reconstructed. In each case, we indicate the metrics and the scales that resulted in the most elongated minimum spanning trees, and as a result, dominated the weighted averages in Eq 4 and 5.

obtained using the Euclidean Distance and KL-Divergence measured over small scales ($l = 4$ and $l = 5$). These scales and metrics dominate the weighted averages in Eq 4 and 5 and, as a result, define the final ordering of the data.

4.2. Examples with real datasets

We now present a series of examples using one-dimensional data. For each dataset, we first display ten objects to convey the typical level of complexity. We then show a visualization of the entire randomly-ordered dataset, followed by the same data this time ordered by the Sequencer. We then display several objects from the ordered dataset. In each case, we indicate the metrics and scales that were identified as the most informative, based on their contributions to the minimum spanning tree elongation-weighted averages in Eq 4 and 5.

Spectroscopic data. We start with a sample of $1,000$ spectra of stars from the publicly-available Sloan Digital Sky Survey (SDSS; York et al. 2000). Each spectrum is a measurement of the brightness of a star as a function of wavelength. The dataset ordered by the Sequencer displays visible trends in both large-scale and small-scales features. A physical interpretation reveals that these continuous variations correspond to a sequence in the temperature of the stars. The top of the sequence is dominated by hot stars, which exhibit absorption lines due to hydrogen atoms, and the bottom part is dominated by cooler stars which exhibit absorption lines due to other, heavier, elements.

We repeat the analysis with a set of $1,000$ spectra of quasars spanning a range of properties. This time we detect a trend in distance (redshift): the bright and narrow features shift continuously throughout the sequence. We point out that, using their default settings, most dimensionality reduction techniques (e.g., PCA, t-SNE, and UMAP) are insensitive to pixel shuffling and often fail to detect horizontal shifts in the data (see section 7.1).

Our third example shows another set of $1,000$ quasar spectra (Trump et al. 2006). The dark pixels correspond to flux deficits due to absorption by gaseous clouds present in front of the background light sources. We point out that this dataset presents a higher level of apparent stochasticity than the previous example and is more difficult to interpret. Here, after reordering the data we also smooth it in the y direction, using a running median filter. this step compensates for the noise, and makes weaker trends more easily apparent. We can observe dark regions corresponding to the absorption of light. The algorithm reveals the existence of two distinct populations of absorbers at the top and bottom of the ordered dataset. This is most obvious when examining the data near $\lambda = 1,500\,\text{Å}$. These systems are known to exhibit different physical properties (e.g., Gibson et al. 2009 and references therein). This example illustrates how the algorithm can naturally perform a clustering task (and define sequences within each cluster), even in the presence of a substantial amount of noise.

Images from the natural world. In the case of simple, highly symmetric physical objects, it is sometimes possible to use a (physical) model and describe each object using a set of parameters. In such a case, it might be possible to identify an underlying trend in the data by looking at a trend in the best fit parameters. When complexity increases, quantitative models based on a small number of parameters are no longer available. The search for trends requires a data-driven approach. To illustrate the performance of the algorithm in a higher-complexity regime where physical modeling of the data is out of reach, we use a set of images from the natural world (natural images). Due to the great variety of shapes and textures, the structural information of such images is expected to be distributed over a wide range of scales and cannot be described by a generic model. To stay in the simpler regime of one-dimensional objects, we randomly shuffle the rows of images and attempt to recover the original ordering. Two examples are shown in Figure 4. The left panel illustrates the complexity of each object and the lack of a simple model to characterize them. For the two examples shown, the algorithm is able to recover the original input. In the Appendix we present more examples using natural images and, for comparison, show the corresponding outputs for t-SNE and UMAP. We point out that, for representational purposes, all re-ordered images are flipped so that they match the expected orientation, when necessary. The algorithm has obviously no information regarding the correct orientation of the output.

Displaying the spatial variation of vectors. Displaying spatial information, for example on a map, is often done through the use of a colorbar, i.e. a sequence. This allows one to display a scalar value as a function of position. If the dynamic range is large, then techniques such as histogram equalization can be used to map the collection of values onto the appropriate range provided by the colorbar. If the quantity to visualize is not a scalar but a vector or a distribution, there is no systematic way to define a mapping onto a colorbar. However, by attempting to order these objects into a sequence, our algorithm provides us with a generic way to meaningfully assign a color to each vector or distribution. The sequencer does to vectors what histogram equalization does to scalars.

To illustrate how the algorithm can be used to display a set of distributions on a map, we apply it to a dataset from structural seismology. Using the surface wave phase velocity at each of about $14,000$ locations across the contiguous United States (Olugboji et al. 2017), we extract the Love wave dispersion curves specifying velocities at a set of 11 periods, which are primarily sensitive to crustal structure. To homog-

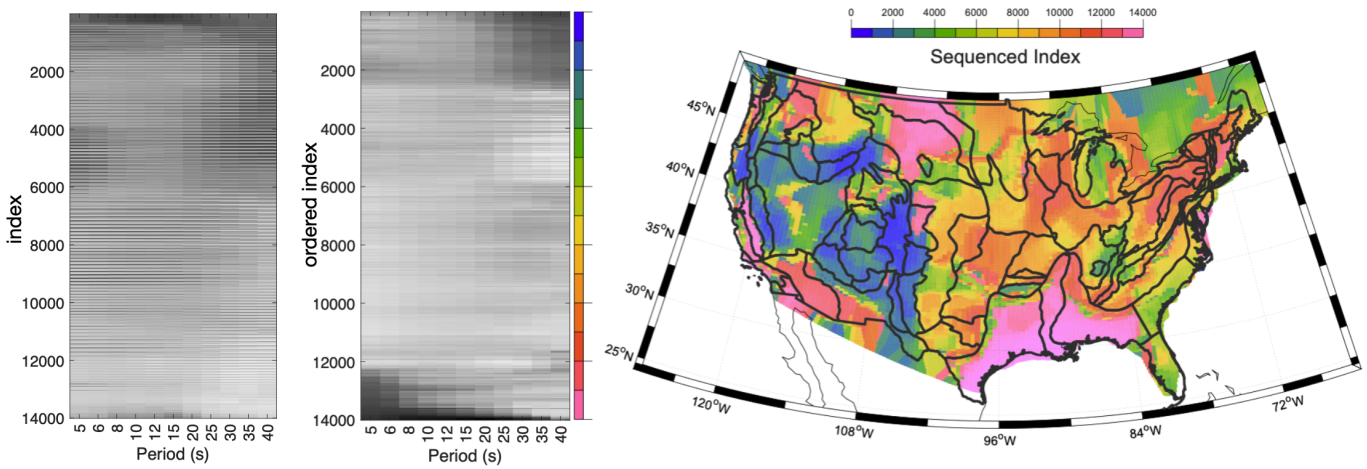

Figure 5. Application of the Sequencer to spatial mapping. Left panels: normalized scores for Love wave phase velocities as a function of period extracted from maps by Olugboji et al. (2017) at about 14,000 locations beneath the contiguous United States and the same data after ordering by the Sequencer. Right panel: locations colored by their sequenced index, revealing geographic patterns which correspond well to physiographic provinces (black outlines, Fenneman 1928). Figure courtesy of Vedran Lekic.

enize the data, we convert the velocity at each period to its standard score. The ordered-index obtained by the Sequencer can be color-coded. Using the same color bar, we can create a map which naturally shows the geographic patterns in crustal structure, shown in Figure 5.

Prominent regions of thick sediment, such as the Williston and Denver basins and the Mississippi Embayment, are traced out at one end of the sequence. Regions with very contrasting crustal structure, such as the Sierra Nevada, High Rockies, and the Snake River Plain, are traced out at the other end of the sequence. We can also point out the correspondence between the geographic patterns revealed by the Sequencer and physiographic provinces identified at the surface (Fenneman 1928), shown by black lines in Figure 5. This example illustrates how the proposed algorithm can be used for mapping properties encoded as distributions rather than scalar values.

5. Discussion

The elongation of the minimum spanning tree of the distance matrix graph characterizing a dataset can be used to identify which aspects of the data, for example which metric and scales, carry the signatures of an underlying simple trend.

Applying the Sequencer algorithm to real data has already led to discoveries. In astrophysics, it revealed a sequence in the spectroscopic properties of Active Galactic Nuclei, which led to a novel way to infer the mass of black holes (Baron & Ménard 2019). In geology, it led to the detection of seismic waves scattered by previously unrecognized 3D structures near the core-mantle boundary (Kim et al. 2020). In each case, the data had been publicly available for years but these trends had not been noticed. Once a trend has been identified by the Sequencer, knowing which observational signatures carry relevant information, it is possible to recover the sequence without the aid of the algorithm.

As the use of the elongation of minimum spanning trees allows one to identify a point of view leading to a simple characterization of the data, it can be used more generically to optimize the parameters of any dimensionality reduction technique (e.g. t-SNE or UMAP, see the Appendix for additional details) in order to create a projection of the data that can reveal the meaningful variation.

Ordering a collection of objects often allows some meaning to emerge. For the datasets shown above, the ordered indices could be assigned to physical quantities: we detected sequences in temperature, distance, type of system, and for natural images we recovered height or angle. Having reached this point, it is the start of the scientific analysis requiring specific domain knowledge. The ability of the Sequencer algorithm to reveal the leading trend in a dataset by analyzing pixel data (rather than selected features or summary statistics) can therefore enable or greatly accelerate scientific discoveries.

5.1. Outlier detection

As mentioned above, if two types of objects are present in a dataset, the algorithm can identify two distinct clusters and will present them sequentially in the final ordering of the data. This will also apply to cases for which a minority of objects, typically labelled as outliers, differ from the rest of the data. In other words, outliers are typically found at one end of the sequence. One can also imagine some objects for which only a small fraction of the pixels differ from the rest of the population. Such objects or set of pixels would also be labelled as outliers. Interestingly, our approach al-

lows us to detect them in a simple manner. Having an ordered sequence in hand, one can meaningfully perform an averaging/smoothing operation along that dimension and reduce the amount of fluctuations not related to the detected sequence. This has the advantage of enabling the detection of weak trends that are not easily visible in the randomly ordered dataset. Then, once a smooth sequence has been defined, one can look at the pixel-level differences between the original sequence and the smooth counterpart. The statistics of these residuals can reveal objects for which a subset of the pixels differ from the expected underlying trend.

5.2. Limitations

A number of considerations must be kept in mind when using the Sequencer: first, the ordering operation performed by the algorithm makes use of one definition of simplicity, based on the elongation of a minimum spanning tree (Eq. 3). While this always provides a well-defined ordering, it might not necessarily lead to the trend expected from model-based considerations, which are often based on a number of assumptions regarding features, scales, metrics, etc. If prior knowledge is available, it is advised to consider limiting the data to the region(s) where a meaningful variation is expected. Similary, in some cases, rescaling the values of the input data might lead to better results, especially in cases for which the data present a high dynamic range.

The algorithm attempts to make use of information on different scales which, by default, are logarithmically spaced. While this approach is meant to be generic, it is possible to imagine cases for which this hierarchical decomposition might not be optimal. Segmenting the data using different strategies (for example using wavelets such that Fourier frequency windows do not overlap) might lead to better results. The user can modify the default sampling strategy if need be.

The algorithm attempts to find a trend using all fluctuations present in the data. These fluctuations can be due to useful structural information or due to random noise. Without a model, the algorithm cannot distinguish between them. Noise fluctuations are partially reduced when averaging is performed across segments of the data and scales (Eq. 4) but do present a limitation in identifying the underlying trend. If the dataset presents a range of noise levels, it is advised to first apply the algorithm to a subset of the data with a higher signal-to-noise ratio. As described above, obtaining an ordered sequence offers another opportunity to perform an averaging operation along the sequence and further improve the characterization of the underlying trend.

6. Conclusions

Exploratory data analysis, i.e. the search for patterns in datasets without the benefit of guidance from theory, is an important part of scientific research. This search is often challenging due to the apparent complexity of the data and, in practice, it tends to be more of an art than a science.

We present an algorithm, the Sequencer, designed to generically find a continuous sequence in a dataset. Using the shape of graphs characterizing similarities between objects, it can identify which aspects of the data carry the signatures of a simple underlying trend. More specifically, given a metric and a scale, it evaluates the degree to which a continuous trend exists based on the elongation of the minimum spanning tree of the corresponding distance matrix. It can then meaningfully combine this information for a collection of metrics and scales to define a final sequence. By extracting only graph-based geometric information, this method allows us to generically identify aspects of the data that lead to a simple description of the underlying variation.

Using the elongation of a minimum spanning tree as a figure of merit can be used in other contexts. For example, it can be used to optimize the parameters of any dimensionality reduction technique in order to create a projection of the data that reveals meaningful variation.

To illustrate the power of the algorithm, we have applied it to various datasets with one dimensional objects. It can straightforwardly be applied to objects in two or higher dimensions, but at a higher computational cost. Using scientific datasets and images with randomly-shuffled rows, we have shown that the Sequencer can identify meaningful trends, even if they originate only from parts of the data. In many cases, it is capable of finding sequences that the popular t-SNE and UMAP dimensionality reduction techniques fail to reveal.

Informed by the geometry of graphs, the algorithm provides guidance in extracting simplicity from observations of the complex world. As already demonstrated in astronomy (Baron & Ménard 2019) and geology (Kim et al. 2020), the Sequencer can discover unexpected (and sometimes simple) trends in datasets that have already been studied by numerous scientists. This approach to exploratory data analysis, which does not rely on any training nor tuning of parameters, has the potential to enable discoveries in a wide range of scientific domains.

Materials and Methods

Code Availability: Implementation details and code are available on GitHub: https://github.com/dalya/Sequencer/. An online interface is available at http://sequencer.org.
The algorithm is implemented in PYTHON and our code relies on the following packages: NUMPY (Oliphant 2006), SCIPY (Jones et al. 2001–), MATPLOTLIB (Hunter 2007), NETWORKX (Hagberg et al. 2008), and SCIKIT-LEARN (Pedregosa et al. 2011). The testing and visualization were done with Jupyter notebooks (Pérez & Granger 2007).

Acknowledgments

We thank Vedran Lekic for his guidance and help with the analysis of the seismology data. We thank Manuchehr Taghizadeh-Popp for the implementation of an online platform running the Sequencer algorithm on uploaded datasets. We thank J. Bloom, J. Kaplan, D. Kim, J. X. Prochaska, Y. S. Ting, and S. Zucker for useful comments on the manuscript. This work was supported by the Packard Foundation and the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. D. Baron is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

7. Algorithm details

7.1. list of metrics

In order to assess the level of similarity between pairs of objects, the algorithm uses a default set of four distance metrics. We briefly describe them here. This set can be expanded by the user if need be. As generically done, we will treat a pair of dataset objects as two probability density distributions $p(i)$ and $q(i)$. An important point to consider is whether the index i encodes a dimension — and can therefore be shuffled, or the value of a coordinate, such as distance, time, energy, etc. — in which case the derivative with respect to i carries valuable information. Distance metrics insensitive to shuffling will not take this into account. When the index i encodes the value of a coordinate, for simplicity, we assume that all objects in the set are sampled in the same manner. If not, interpolation/resampling techniques need to be applied prior to using the Sequencer algorithm. The four metrics used by default are:

- **The Euclidean Distance (L_2)**, i.e. the familiar distance between two points in an Euclidean space, is the default metric used in many fields. In particular, it is the default metric used by the dimensionality reduction algorithms t-SNE and UMAP. Given two objects that are represented by the vectors $p(i)$ and $q(i)$, L_2 is given by:

$$\mathrm{L}_2(p,q) = \sqrt{\sum_i \Big[p(i) - q(i)\Big]^2}. \tag{6}$$

  L_2 is non-negative, symmetric, and equals to zero only when the two vectors are identical. Furthermore, L_2 is insensitive to the relative order of the values in the vector and therefore does not use any (useful) information from the derivatives with respect to index i.

- **The Kullback-Leibler Divergence (KL-divergence)**, also called the relative entropy, quantifies the degree of surprise involved in seeing one distribution, given another one. It is widely used in Machine Learning to compare the similarity between two objects. For two discrete random variables with PDFs $p(i)$ and $q(i)$, it is given by:

$$\mathrm{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)} \tag{7}$$

  The KL-divergence is non-negative, and equals to zero only when the two functions are similar in every bin i. One can see that it is asymmetric, namely $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$, and it reaches infinity if in at least one bin i, $q(i) = 0$. We further note that the KL-divergence is not sensitive to the relative order of the bins.

- **The Monge-Wasserstein or Earth Mover Distance (EMD):** measures a distance between two distributions from an optimal transport point of view. Intuitively, if the distributions are interpreted as two different ways of piling up a given amount of dirt, the EMD is the minimum work required to turn one pile of dirt into the other, where work is the amount (mass) of dirt moved times the distance by which it moved. The EMD is widely used in computer vision, and specifically in content-based image retrieval, as it resembles remarkably well human's visual perception (see e.g., Rubner et al. 2000). Interestingly, for one-dimensional distributions, the solution of the optimal transport can be obtained simply by (Ramdas et al. 2015):

$$\mathrm{EMD}(p,q) = \int_{-\infty}^{\infty} |P(i) - Q(i)| di, \tag{8}$$

where $P(i)$ and $Q(i)$ are the cumulative distribution functions of the random variables $p(i)$ and $q(i)$. For higher dimensional distributions, computing the EMD is more computationally demanding. The EMD is non-negative and symmetric. In contrast to L_2 and the KL-divergence, the EMD is sensitive to the order and the value of the indices. It can therefore track and quantify translations and/or derivatives with respect to i. As a result, the EMD can capture certain aspects of similarities between objects to which L_2 and the KL-divergence might not be sensitive.

- **The Energy Distance (ED)** provides another measure of statistical distance between two probability distributions. Székely (2002) showed that for one-dimensional real-valued random variables, $p(i)$ and $q(i)$, with cumulative distribution functions $P(i)$ and $Q(i)$, it is equivalent to:

$$\mathrm{ED}(p,q) = \left(2\int_{-\infty}^{\infty} |P(i) - Q(i)|\right)^2 di. \tag{9}$$

The ED is non-negative and symmetric. Similarly to the EMD, it is sensitive to the order and values of the indices. The differences between the ED and the EMD are analogous to the differences between the L_1 and L_2 norms.

7.2. Graph nomenclature

Here we provide a visual support to summarize the key quantities used by the algorithm and the corresponding terminology introduced in section 3. Figure 6 shows the example of a minimum spanning tree, where the nodes are color-coded according to their centrality measure. The least connected node of this graph is indicated in the bottom left of the figure. It provides the starting point to traverse the graph using a Bread First Search (BFS) walk, i.e. along the longest branch of the graph and scanning each branch along the way. The nodes are ordered in levels according to their distance (in units of edges) from the starting point (marked with numerical labels in the figure). These levels are used to estimate the average width and height of the tree, which are then used to estimate the elongation of the minimum spanning tree. The graph length (Eq. 1) and width (Eq. 2) are also illustrated in the figure.

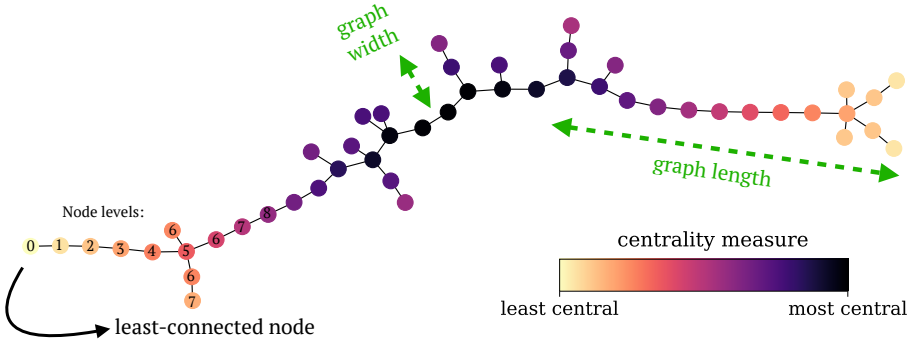

Figure 6. Illustration of various graph minimum spanning tree properties used by the algorithm.

7.3. Faster computation for large datasets

The algorithm described in section 3 requires distance matrix calculations which scale as $\mathcal{O}(N^2)$. For large datasets this approach can become too computationally demanding. A useful alternative is to first apply the technique to a subset of the data, build the skeleton of a sequence and then populate it with the remaining points. This process leads to a faster computation and enables the analysis of much larger datasets. It can be implemented as follows: for a dataset with N objects, we first select a subset with $N_s \ll N$ objects for which the distance matrix calculations are computationally feasible. This provides us with a first sequence.

- **growing sequence:** given a selected sequence, the algorithm selects a fraction f_A of objects distributed uniformly. We refer to them as anchor points.

- **adding new objects:** to populate the growing sequence with a new object i, we first perform a low-resolution search where we measure the distance between this object to the $N_A = f_A\, N_s$ anchor points and find the two nearest neighbors. We then perform a high-resolution search computing distances between object i and all the nodes from the evolving sequence

located between the two nearest anchor points. These distances are populated into the proximity matrix, with a weight that corresponds to the aggregated minimum spanning tree elongation for a given scale (Eq. 5). This proximity matrix is converted into a distance matrix, after which a minimum spanning tree is constructed. We note that, this part of the calculation involves very sparse matrices as only a very small fraction of nodes and edges are considered. This process of populating the growing sequence can be done in parallel for a group of objects. Finally, the updated growing sequence is obtained by a Breadth First Search traversal of the minimum spanning tree. This concludes a single iteration of the population phase.

- **Updated sequence:** after having populated the growing sequence with a group of new objects, we obtain an updated sequence. We can then repeat the process with the remaining points. Here we point out that, at each iteration, the number of *anchor* points, defined to be a fraction f_A of the size of the coarse sequence, grows linearly with the size of the growing sequence. In order to ensure convergence, the number of objects that are populated in every iteration must be smaller than the number of anchor points.

This process avoids a full $\mathcal{O}(N^2)$ calculation and allows one to search for sequences in large datasets in a more efficient manner. This is done at the cost of obtaining only an approximate result for which the accuracy depends on the choice of initial subset size N_s and the fraction f_A of anchor points to use.

8. Minimum Spanning Tree elongation as a figure of merit

One of the key ideas presented in this work is that it is possible to quantify the level to which a trend is detectable in a dataset. One can do so by using the elongation of the minimum spanning tree of a distance matrix characterizing this dataset or a collection of distance matrices characterizing various aspects of the dataset. In order to use this method generically and, for example, to compare performances with datasets of different sizes, we can use a normalized elongation parameter:

$$\eta' = \eta / N \tag{10}$$

where N is the number of objects in a dataset. This normalized elongation ranges from $1/N$ for a random graph to 1 for a perfect sequence. This normalized elongation parameter can be computed for any dataset and for the output of any dimensionality reduction technique. It can thus serve as a figure of merit to quantify the level to which a trend is detectable. It can therefore be used to optimize the parameters of a given algorithm and/or select which technique performs better in order to reveal a continuous trend in a dataset. This procedure could for example be used in combination with the t-SNE and UMAP algorithm to select their internal parameters, metric and possibly the scales of the data leading to the embedding revealing the "best" sequence.

8.1. Comparing the Sequencer to t-SNE and UMAP

In order to compare the performance of the Sequencer to that of t-SNE or UMAP,

we need to introduce a procedure to automatically select their parameters leading to the "best" sequence but without using the full machinery of the Sequencer. To do so, we proceed as follows: given a set of hyper-parameters, each of these techniques can be applied to embed a collection of objects or vectors into a one-dimensional space and naturally obtained an ordered list. The question then is whether this list reveals a meaningful trend in the dataset. To address this, it is unfortunately not possible to directly use the elongation parameter in that embedding as, by definition, the corresponding manifold only has one dimension, so η' is always one. To meaningfully estimate a corresponding elongation parameter, we need to have access to more information. To do so, we use these embedding techniques to obtain projections in two dimensions. In the corresponding spaces, we can now meaningfully estimate the elongation of these manifolds and use this information as a figure of merit to assess the level to which a sequence is found. Here, we point out that the elongation parameter estimated in this manner can serve as a figure of merit only when the two-dimensional embedding does not strongly depart from a one-dimensional manifold. It is meaningfully defined only when the two-dimensional distribution of objects can be simply represented by a major and minor axis. In cases involving clusters and/or outliers in the two-dimensional projection, the elongation parameter no longer informs on the quality of an expected sequence.

As described above, we can then use the elongation parameter as a figure of merit to find the metric and/or hyper-parameters that produce the most elongated sequence or, in other words, the trend presenting the most elongated one-dimensional manifold, i.e. the highest degree of continuity. To illustrate this, we apply t-SNE and UMAP to two images for which the rows have been randomly shuffled. In Figure 7 we show how varying the hyper-parameters of each technique affects the quality of the resulting embedding and how optimizing for the largest value of the elongation parameter naturally leads to the recovery of the original

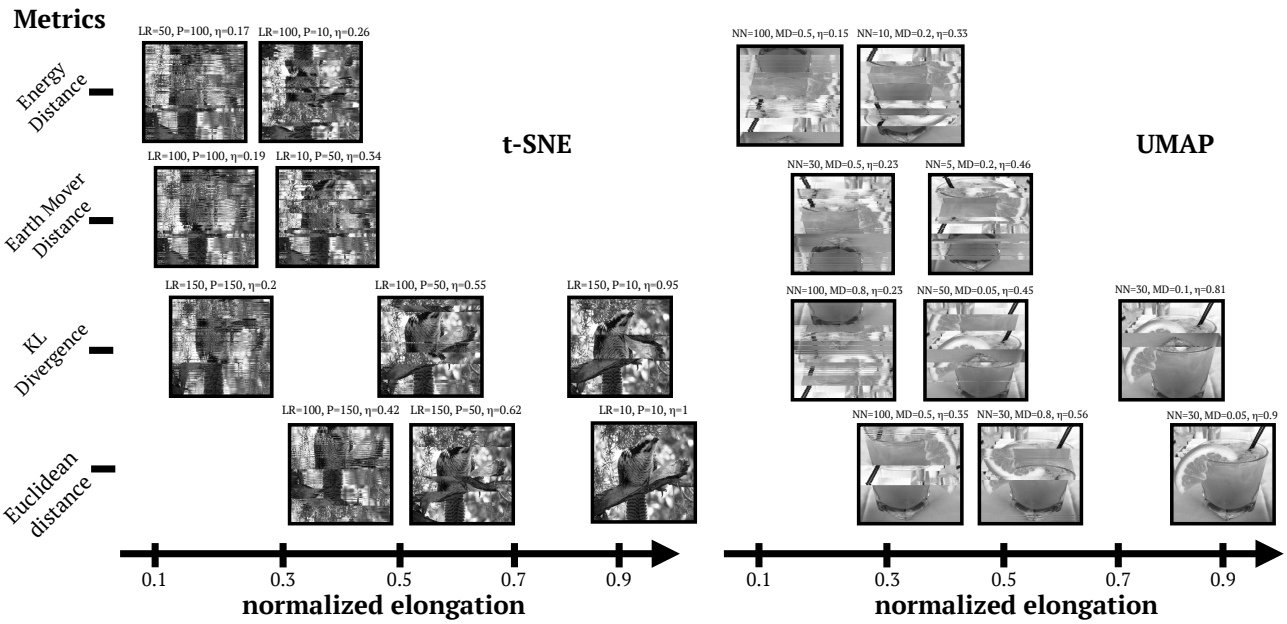

Figure 7. Using the elongation parameter as a figure of merit to optimize the hyper-parameters and/or metric of t-SNE and UMAP. These two embedding techniques are applied to randomly shuffled rows of images. The highest minimum spanning tree elongations naturally selects the correct ordering of the objects within the set.

image. We also point out that optimizing the hyper-parameters of these techniques does not always allow to recover the original image.

Having introduced the normalized elongation parameter to (i) automatically optimize the hyper-parameters and/or metrics of t-SNE/UMAP and (ii) compare the resulting embeddings of different techniques, we now show that the Sequencer algorithm can identify an underlying trend in cases where both t-SNE and UMAP do not. This is similar to the example shown in Section 8 but, this time, we are using real data rather than artificially generated distributions. To illustrate this point, we select images from the COCO dataset (Lin et al. 2014). As done previously, we treat the rows as a collection of ordered objects whose order is randomly shuffled. We use them as inputs to the Sequencer, t-SNE and UMAP, and in each case we attempt to recover the original ordering using the procedure described above. For t-SNE, we consider the four distance metrics and the hyper parameters: `learning_rate=[10, 50, 100, 150]` and `perplexity=[10, 50, 70, 100]`. For UMAP, we consider the same distance metrics and the hyper-parameters: `n_neighbour=[10, 50, 100, 150]` and `min_dist=[0.1, 0.2, 0.5, 0.8]`. We note that this coarse sampling of the hyper parameters of these two techniques appears to be enough for this type of data. A higher resolution optimization of the parameters does not provide better results.

The results are shown in Figure 8. For each input shuffled image, we present the output of the Sequencer, as well as the ”best” outputs obtained with t-SNE and UMAP. In all cases, the elongation-based optimization leads to meaningful segments of the original images. Interestingly, in a number of cases, the Sequencer outperforms the best possible outputs obtained with t-SNE and UMAP. As described in section 4.1, this is due to the ability of the Sequencer to look for a global trend using multiple metrics and a range of scales.

References

Baron, D. 2019, arXiv e-prints, arXiv:1904.07248. https://arxiv.org/abs/1904.07248

Baron, D., & Ménard, B. 2019, MNRAS, 487, 3404, doi: 10.1093/mnras/stz1546

Carlsson, G. 2009, Bulletin of the American Mathematical Society, 46, 255

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. 2009, Introduction to Algorithms, Vol. 1 (MIT press)

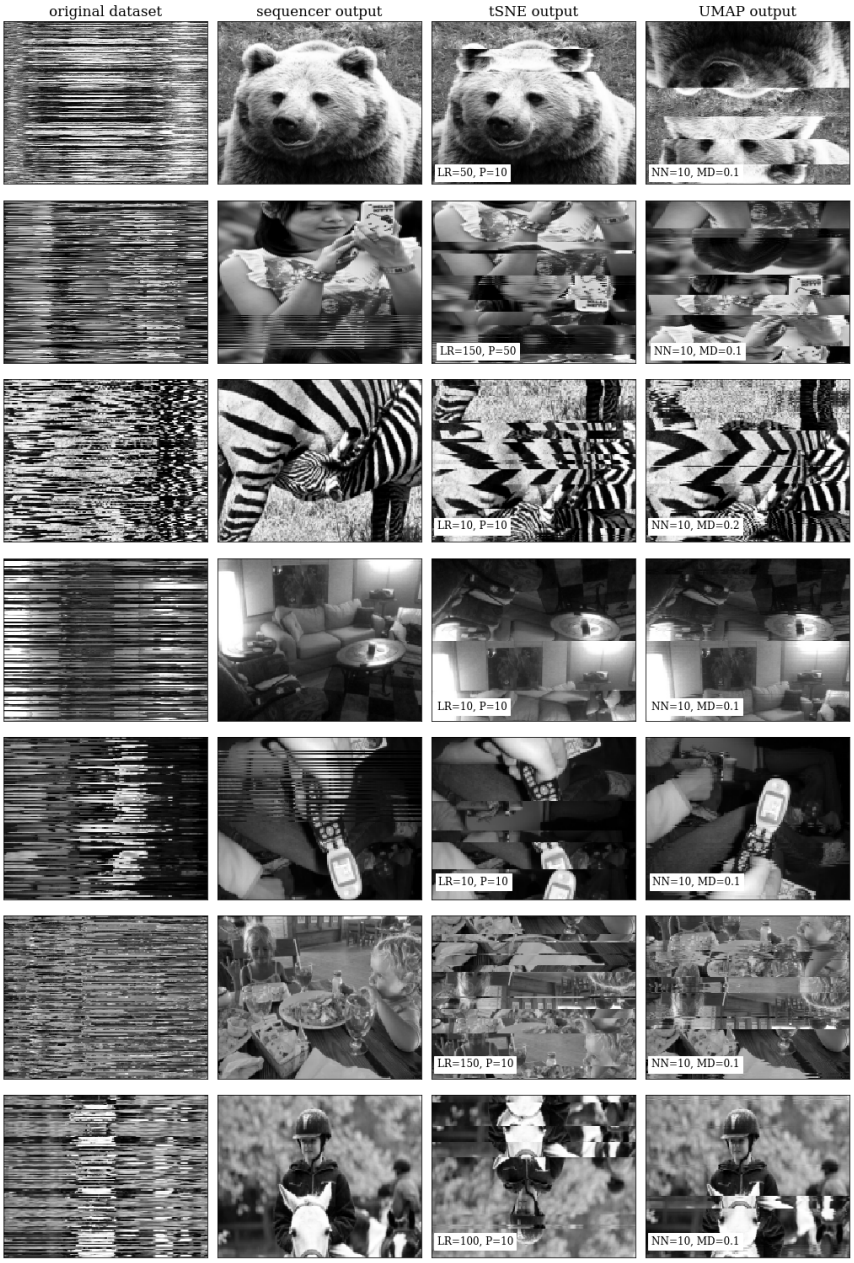

Figure 8. Comparison between the Sequencer output and the one-dimensional embedding by t-SNE and UMAP for scrambled images. The first column represents the original scrambled images used as inputs to the three algorithms. The second column shows the reordered image according to the Sequencer's output. The third and forth rows show the reordered image according to the "best" one-dimensional representation by t-SNE and UMAP. For t-SNE and UMAP, at the top of each image we indicate the hyper-parameters that resulted in the highest elongation parameter.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. 1996, in KDD, ed. E. Simoudis, J. Han, & U. M. Fayyad (AAAI Press), 226–231. http://dblp.uni-trier.de/db/conf/kdd/kdd96.html#EsterKSX96
Fenneman, N. M. 1928, Annals of the Association of American Geographers, 18, 261
Freeman, L. C., & Freeman, L. C. 1978, SOCIAL NETWORKS, 215, doi: 10.1.1.227.9549
Gibson, R. R., Jiang, L., Brandt, W. N., et al. 2009, ApJ, 692, 758, doi: 10.1088/0004-637X/692/1/758
Hagberg, A. A., Schult, D. A., & Swart, P. J. 2008, in Proceedings of the 7th Python in Science Conference, ed. G. Varoquaux, T. Vaught, & J. Millman, Pasadena, CA USA, 11 – 15. http://conference.scipy.org/proceedings/SciPy2008/paper_2/
Hunter, J. D. 2007, Computing In Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55
Jones, E., Oliphant, T., Peterson, P., et al. 2001–, SciPy: Open source scientific tools for Python. http://www.scipy.org/
Kim, D., Lekic, V., Ménard, B., Baron, D., & Taghizadeh-Popp, M. 2020, Submitted to Science
Kruskal, J. B. 1956, Proceedings of the American Mathematical Society, 7, 48
Kullback, S., & Leibler, R. A. 1951, Ann. Math. Statist., 22, 79, doi: 10.1214/aoms/1177729694
Lee, J. A., & Verleysen, M. 2010, Pattern Recognition Letters, 31, 2248 , doi: https://doi.org/10.1016/j.patrec.2010.04.013
Lin, T.-Y., Maire, M., Belongie, S., et al. 2014, arXiv e-prints, arXiv:1405.0312. https://arxiv.org/abs/1405.0312
MacQueen, J. B. 1967, in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, ed. L. M. L. Cam & J. Neyman, Vol. 1 (University of California Press), 281–297
McInnes, L., Healy, J., & Melville, J. 2018, arXiv e-prints, arXiv:1802.03426. https://arxiv.org/abs/1802.03426
Oliphant, T. E. 2006, A guide to NumPy, Vol. 1 (Trelgol Publishing USA)
Olugboji, T., Lekic, V., & McDonough, W. 2017, Tectonics, 36, 1232
Pearson, K. 1901, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2, 559, doi: 10.1080/14786440109462720
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825
Pérez, F., & Granger, B. E. 2007, Computing in Science and Engineering, 9, 21, doi: 10.1109/MCSE.2007.53
Ramdas, A., Garcia, N., & Cuturi, M. 2015, arXiv e-prints, arXiv:1509.02237. https://arxiv.org/abs/1509.02237
Rodriguez, A., & Laio, A. 2014, Science, 344, 1492, doi: 10.1126/science.1242072
Rubner, Y., Tomasi, C., & Guibas, L. J. 1998, in Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), 59–66
Rubner, Y., Tomasi, C., & Guibas, L. J. 2000, International Journal of Computer Vision, 40, 99, doi: 10.1023/A:1026543900054
Székely, G. J. 2002, Bowling Green State University, Department of Mathematics and Statistics, Technical Report 02-16
Trump, J. R., Hall, P. B., Reichard, T. A., et al. 2006, ApJS, 165, 1, doi: 10.1086/503834
van der Maaten, L., & Hinton, G. 2008, Journal of Machine Learning Research, 9, 2579
Van Der Maaten, L., Postma, E., & Van den Herik, J. 2009, J Mach Learn Res, 10, 66
Venna, J., Kaski, S., Peltonen, J., Nybo, K., & Aidos, H. 2010, Journal of Machine Learning Research, 11, 451
Ward, J. H. 1963, Journal of the American Statistical Association, 58, 236, doi: 10.1080/01621459.1963.10500845
Wattenberg, M., Viégas, F., & Johnson, I. 2016, Distill, doi: 10.23915/distill.00002
Yizong Cheng. 1995, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17, 790, doi: 10.1109/34.400568
York, D. G., Adelman, J., Anderson, Jr., J. E., et al. 2000, AJ, 120, 1579, doi: 10.1086/301513
Zhang, P., Ren, Y., & Zhang, B. 2011, arXiv e-prints, arXiv:1108.1636. https://arxiv.org/abs/1108.1636