

Hypothesis and analysis on the relationship between bivariate data correlation and Mapper graph structure

Phillip Korolev

January 2026

1 Introduction

When tasked with analyzing data, we are often looking to draw intuitive, visual patterns. With bivariate time series data, our task is much simpler and typically narrows our choice for data analysis techniques. In this case, the goal is to draw some sort of pattern between our two observed values $X(t)$, $Y(t)$ over a portion or the entirety of our data set's time span T . Although a 2-dimensional scatter plot is often our first step to visualize some sort of correlation between the values, it can be difficult to tell just how populated clusters of points are, or to draw conclusions given unintuitive plot shapes.

With methods in topological data analysis, we are granted techniques that let us explore the structure and correlation of our data past the initial scatter plot. In this informal analysis, we will introduce the Mapper algorithm, which transforms arbitrary dimensional data into a flat discrete graph with nodes, edges, and coloring, helping us identify things like cluster density and potential cycles in our time series data. In particular, we will analyze the structure of graphs generated from highly correlated and uncorrelated bivariate time series data sets.

2 Mapper Algorithm

Many more detailed versions of the algorithm and motivations can be found in outside resources. Though, the steps relevant to our experiment will be highlighted in the second algorithm rundown.

Simply, the Mapper algorithm uses filtration processes (of our choice) to project data $X \subset \mathbb{R}^d$ onto \mathbb{R}^m with $m \leq d$. After doing so, we initialize overlapping covers of the space to split and group the projected image points into their respective covers. Classifying our original pre-image points (in \mathbb{R}^d), we then apply a clustering method onto each cover of points. After classifying points and applying clustering, we are able to construct a graph where each node represents some cluster, and an edge is only present if there are overlapping points between two clusters.

Formally, we write the general algorithm as such:

1. Normalize our data set $X \subset \mathbb{R}^d$ using a normalization function of choice $\text{Norm} : X \rightarrow \tilde{X}$
2. Choose a filtration (lens) function $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$, typically with $m \in \{1, 2\}$, and build the image set $f(\tilde{X}) = \{f(x) : x \in \tilde{X}\}$

3. In the image space, construct a cover $\mathcal{U}_f = \{U_a\}_{a \in A}$, with A an arbitrary index set, such that:
 - (i) their union covers all of the image space: $f(\tilde{X}) \subseteq \bigcup U_a$
 - (ii) for $a, b \in A$: U_a, U_b adjacent $\iff U_a \cap U_b \neq \emptyset$
4. Classify points $x \in \tilde{X}$ with class c if $f(x) \in U_c$
5. Apply clustering to each group of points by class
6. Construct the graph $G = (V, E)$ with vertices v_i representing each cluster and edges connecting v_i, v_j if and only if there exists a point $x \in \tilde{X}$ such that $x \in C_i \cap C_j$

In the case of the experiments of this paper, the algorithm can be slightly more defined. To start, we are interested in bivariate data. That is, our data set X is 2-dimensional and of the form

$$X = \{(x_i, y_i)\},$$

so already, we are restricted to using filtration functions in $\mathcal{F} = \{f \mid f : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \cup f : \mathbb{R}^2 \rightarrow \mathbb{R}\}$. In most cases, and in the experiments seen in further sections, we use the filtration function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ defined by

$$f(x, y) = |x - y|,$$

to stay on the theme of correlation. Given that \tilde{X} is the normalized data set, the function f above gives us a good idea of how far apart the differential between the values \tilde{x}_i and \tilde{y}_i is. Then, since the image $f(\tilde{X})$ lies on the real number line, our collection of overlapping covers is just a collection of intervals I_1, I_2, \dots, I_N satisfying:

$$\bigcup_{i=1}^N I_i = [\min f(\tilde{X}), \max f(\tilde{X})].$$

It is also important to mention that we ensure some overlap between these intervals. That is, we set an overlap parameter $\alpha \in (0, 1)$ that tells the interval construction algorithm the amount of overlap that should occur between each interval. Note that for any two consecutive intervals I_k, I_{k+1} we have that $I_k \cap I_{k+1} = [\min I_{k+1}, \max I_k]$, with the guarantee of overlap, this intersection is never empty.

Example. Take the interval $\mathcal{I} = [0, 10]$, we build $N = 3$ equally wide intervals with overlap $\alpha = 0.5$ such that I_1, I_2, I_3 cover all of \mathcal{I} . We want to do so "efficiently", so our first interval I_1 should begin at $\min \mathcal{I} = 0$, and $I_N = I_3$ should have maximum $\max \mathcal{I} = 10$. This algorithm is described below, but for the sake of the example, our covers would be the following:

$$I_1 = [0, 5]; I_2 = [2.5, 7.5]; I_3 = [5, 10],$$

with intersections $I_1 \cap I_2 = [2.5, 5] = [\min I_2, \max I_1]$ and $I_2 \cap I_3 = [5, 7.5] = [\min I_3, \max I_2]$.

The following is the algorithm used to generate graphs and results seen in this paper. As described above, it is tuned for exploring correlation structures in bivariate data. Also, the normalization method used is Z -normalization, unless stated otherwise, which we will denote $\text{Norm}_Z : X \subset \mathbb{R}^2 \rightarrow \mathbb{R}^2$ and is defined by:

$$\text{Norm}_Z(x, y) = \left(\frac{x - \mu_x}{\sigma_x}, \frac{y - \mu_y}{\sigma_y} \right)$$

with μ_x and σ_x the mean and standard deviation with respect to the x argument in X . Likewise for μ_y , σ_y .

Algorithm 1: Mapper Construction with Absolute-Difference Filtration

Input: Data set $X \subset \mathbb{R}^2$, number of intervals N , cover overlap parameter α
Output: Nerve graph G

1. $\tilde{X} \leftarrow \text{Norm}_z(X)$
2. Define function $f(x, y) = |x - y|$
3. $Y \leftarrow f(\tilde{X})$
4. $\mathcal{I} \leftarrow [\min f(\tilde{X}), \max f(\tilde{X})]$
5. $\mathcal{U} \leftarrow \text{getCovers}(\mathcal{I}, N, \alpha)$
6. **foreach** $I_i \in \mathcal{U}$ **do**
 - Classify points $c_i \leftarrow \{f(x) \in Y : f(x) \in I_i\}$
 - Pull back $V_i \leftarrow f^{-1}(I_i)$
 - Compute clusters $\mathcal{C}_i \leftarrow \mathcal{C}(V_i)$
7. Construct graph G :
 - (i) add node N_i representing cluster \mathcal{C}_i
 - (ii) add edge (N_i, N_j) if and only if $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$ for $i \neq j$

return G

For the cover construction process described previously, and seen in Algorithm 1, we use the following. Note that the algorithm constructs intervals as we assume that all filtration functions used throughout the experiments in the paper are of the form $f : \mathbb{R}^2 \rightarrow \mathbb{R}$.

Algorithm 2: Interval Construction in \mathbb{R} (used for getCover above)

Input: Interval $\mathcal{I} \subset \mathbb{R}$, number of intervals N , overlap parameter α
Output: List of intervals I_1, \dots, I_N covering \mathcal{I}

1. Initialize list L
2. $w \leftarrow (\max \mathcal{I} - \min \mathcal{I}) / (N - (N - 1)\alpha)$
3. $s \leftarrow w(1 - \alpha)$
4. **for** $i \in 0, 1, \dots, N - 1$ **do**
 - $l \leftarrow \min \mathcal{I} + i \cdot s$
 - $u \leftarrow l + w$
 - $L \leftarrow I_i = [l, u]$

return L

3 Mapper Process Visualized

Given a sample bivariate data set $X = \{(x_i, y_i)\}$, we normalize and plot the data using Norm_Z as described previously:

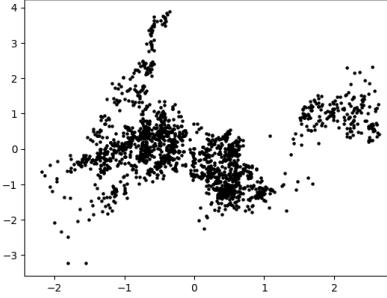


Figure 1: Scatter of normalized sample data

We can already tell that our data is highly correlated, but we proceed with example as if this fact is irrelevant. We then use the filtration $f(x, y) = |x - y|$ to project the data onto \mathbb{R} :

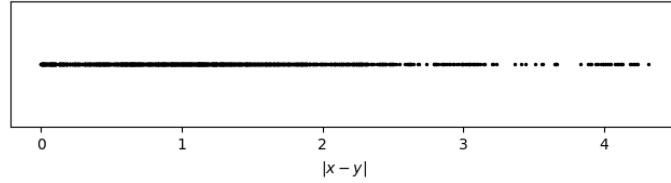


Figure 2: Projection of X onto \mathbb{R} using filtration f

Then, building $N = 3$ covers (intervals) using overlap $\alpha = 0.3$, we label each cover by color red, green, and blue respectively. We then classify the projected points $f(\tilde{X})$ by their respective cover, and if some point $f(d) \in f(\tilde{X})$ belongs to two covers I_i and I_j , we simply classify it as a mixed color. That is, if class I_1 is red, and I_2 is green, a point $f(d) \in I_1 \cap I_2$ will be classified yellow.

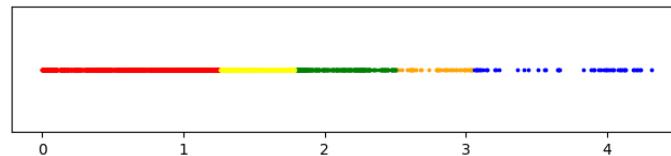


Figure 3: Projection points classified by cover