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

Phillip Korolev

January 2026

1 Introduction

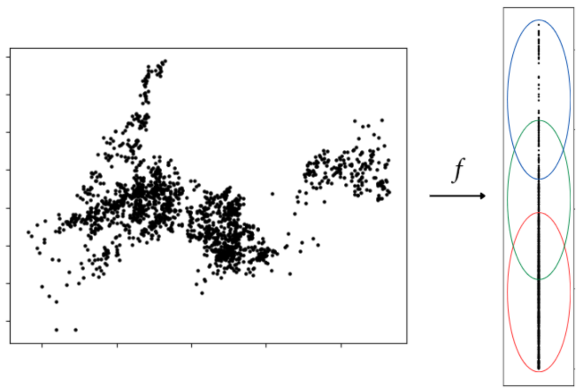
When tasked with analyzing data, we are often looking to draw intuitive, visual patterns. With bivariate time series data, our task is much simpler and typically narrows our available options for data analysis techniques. In this case, the goal is to draw some sort of pattern between our two observed 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. The general summary of the algorithm will be presented, as well as a more formal process.

Simply, the Mapper algorithm uses filtration processes (of our choice) to project data $X \subset \mathbb{R}^d$ onto $\mathbb{R}^m$ with $m < d$. The filtration function is often called the lens, and typically we choose a function that projects our data onto $\mathbb{R}^2$ or $\mathbb{R}$. Based on given parameters, we look to construct overlapping covers of the lens space $f(X)$ to split and group the projected image points into their respective regime.



The colored portions of our lens space $f(X)$ above represent a different cover (interval) of the space and help us classify our original points $f^{-1}(X)$. After classifying the original points and applying some clustering algorithm, 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. Also note that the size of a particular node depends on the respective cluster's density (population).

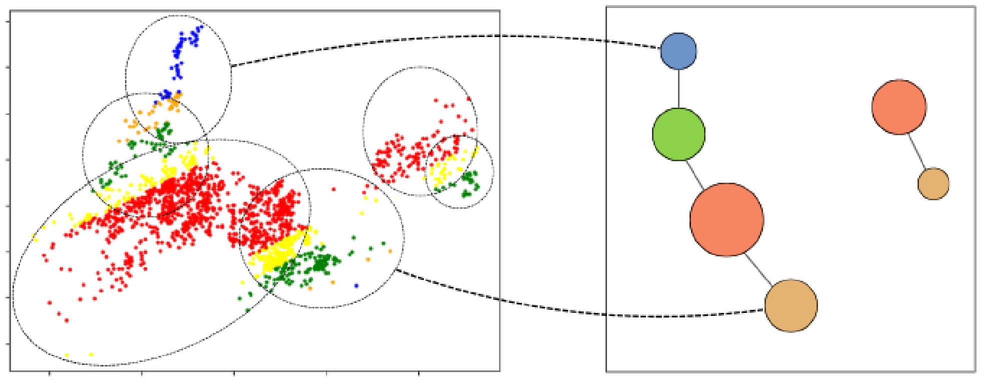


Figure 1: Clustering classified data points and generating graph

Generally, the process aims to accomplish the following:

1. Normalize our data set $X \subset \mathbb{R}^d$ using a normalization function of choice $\text{Norm} : X \to \tilde{X}$
2. Choose a filtration (lens) function $f : \mathbb{R}^d \to \mathbb{R}^m$, 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 our data $\tilde{X}$
6. Construct the graph $G = (V, E)$ with vertices v_i representing clusters C_i and edges e_i connecting two vertices 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 of the form $X = \{(x_i, y_i)\}$, so the normalization we use is the process $\text{Norm}_Z : X \to \tilde{X}$, 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.

With X as described above, we are restricted to using filtration functions in $\mathcal{F} = \{f \mid f : \mathbb{R}^2 \to \mathbb{R}^2 \cup f : \mathbb{R}^2 \to \mathbb{R}\}$. In the experiments seen in further sections, we use the filtration function $f : \mathbb{R}^2 \to \mathbb{R}$ 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 values $\tilde{x}_i$ and $\tilde{y}_i$ are. For some intuition, a perfectly correlated normalized dataset $\tilde{X}$ would lie on the line $y = x$ in $\mathbb{R}^2$. In our case, the image space $f(\tilde{X})$ lies on the real number line, so our collection of overlapping covers is just a collection of overlapping intervals $I_1, I_2, ..., I_N$ satisfying:

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

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

Example. Interval Construction
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.

Algorithm 1: Mapper Construction with Absolute-Difference Filtration

```
Input: Data set $X \subset \mathbb{R}^2$, number of intervals $N$, cover overlap parameter $\alpha$
Output: Nerve graph $G$
1. $\tilde{X} \leftarrow \mathrm{Norm}_z(X)$
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. $\mathcal{U} \leftarrow \mathrm{getCovers}(\mathcal{I}, N, \alpha)$
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 (**getCovers**), 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 $f : \mathbb{R}^2 \to \mathbb{R}$.

Algorithm 2: Interval Construction in $\mathbb{R}$

```
Input: Interval $\mathcal{I} \subset \mathbb{R}$, number of intervals $N$, overlap parameter $\alpha$
Output: List of intervals $I_1, ..., I_N$ 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 $i \in 0, 1, ..., N-1$ 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:

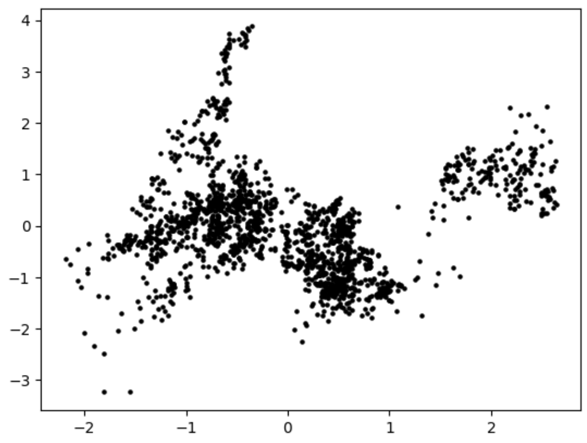


Figure 2: Scatter of normalized sample data

Our scatter creates a nontrivial shape, but we continue to capture this distance between normalized points $d(\tilde{x}_i, \tilde{y}_i)$. Using filtration function $f(x, y) = |x - y|$, we project the data onto $\mathbb{R}$:

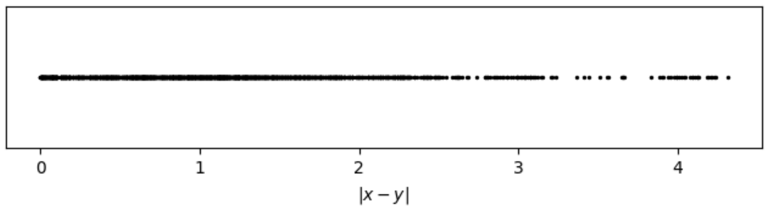


Figure 3: Projection of X onto $\mathbb{R}$ using filtration f

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.

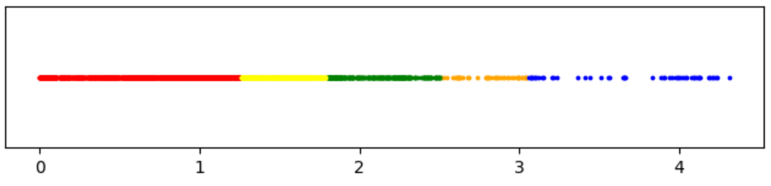


Figure 4: Projection points classified by cover

Bringing our same classification rule back into the original plot of X, we use the following procedure: classify $(x, y) \in X$ as class c_i if $f(x, y) \in I_i$. Again, we have the dominating intersection rule that states if $f(x, y) \in I_i \cap I_2$, then classify (x, y) as the mixed class c_{ij}. In the context of this example, if $f(x, y) \in I_1$ strictly, then, classify (x, y) as class c_1 (red). Likewise, if some point $f(x, y) \in I_1 \cap I_2$, classify (x, y) as the mixed class c_{12} (yellow). The following is the plot X after pulling back classification:

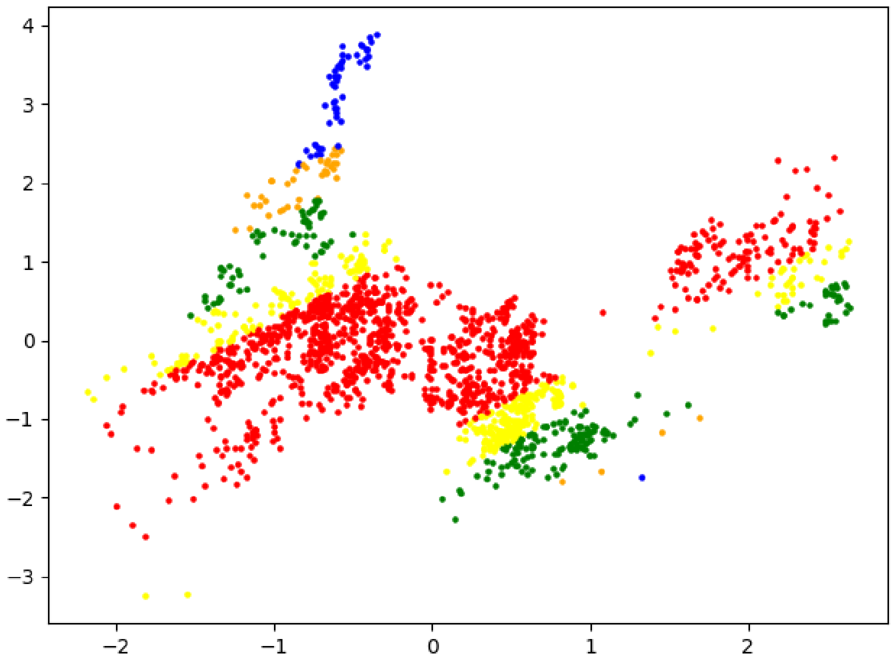


Figure 5: Plot of X post classification

Since we are using the absolute difference as our lens function (filtration function), we are essentially splitting our pre-image space into portions $y = x \pm \delta$ for some $\delta \in \mathbb{R}$. Consider the set of points $\{(p, q) \in \mathbb{R}^2 : 0 < |p - q| < 1\}$. This, for example, would be its own class, given our lens function was the absolute difference and appropriate intervals were chosen. We can think of red points above (points in class c_1) as points with absolute difference in argument of no more than roughly 1.8 (approximated from Figure 3).