Finding Patterns with a Rotten Core: Data Mining for Crime Series with Cores

As Published: 10.1089/BIG.2014.0021

Publisher: Mary Ann Liebert Inc

Persistent URL: https://hdl.handle.net/1721.1/133897

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Big Data
Volume 3 Number 1, 2015
Mary Ann Liebert, Inc.
DOI: 10.1089/big.2014.0021

ORIGINAL ARTICLE

Finding Patterns with a Rotten Core: Data Mining for Crime Series with Cores

Tong Wang,[1] Cynthia Rudin,[1,*] Daniel Wagner,[2] and Rich Sevieri[2]

Abstract

One of the most challenging problems facing crime analysts is that of identifying *crime series*, which are sets of crimes committed by the same individual or group. Detecting crime series can be an important step in predictive policing, as knowledge of a pattern can be of paramount importance toward finding the offenders or stopping the pattern. Currently, crime analysts detect crime series manually; our goal is to assist them by providing automated tools for discovering crime series from within a database of crimes. Our approach relies on a key hypothesis that each crime series possesses at least one *core* of crimes that are very similar to each other, which can be used to characterize the modus operandi (M.O.) of the criminal. Based on this assumption, as long as we find all of the cores in the database, we have found a piece of each crime series. We propose a subspace clustering method, where the subspace is the M.O. of the series. The method has three steps: We first construct a similarity graph to link crimes that are generally similar, second we find cores of crime using an integer linear programming approach, and third we construct the rest of the crime series by merging cores to form the full crime series. To judge whether a set of crimes is indeed a core, we consider both *pattern-general similarity*, which can be learned from past crime series, and *pattern-specific similarity*, which is specific to the M.O. of the series and cannot be learned. Our method can be used for general pattern detection beyond crime series detection, as cores exist for patterns in many domains.

Key words: crime series detection; subspace clustering; clustering with feature selection; pattern mining; dense core sets; similarity graph

Introduction

One of the most important problems in crime analysis is that of *crime series detection*, or the detection of a set of crimes committed by the same individual or group. Criminals follow a modus operandi (M.O.) that characterizes their crime series; for instance, some criminals operate exclusively during the day, others work at night, some criminals target apartments for housebreaks, while others target single family houses. Crime analysts need to identify these crime series within police databases at the same time as they identify the M.O. Currently, analysts identify crime series by hand: They manually search through the data using database queries trying to locate patterns, which (as Ref. notes) can be very challenging and time-consuming. From a computational perspective, crime series detection is a very difficult problem: It is a clustering problem with cluster-specific feature selection, where the set of features for the cluster is the M.O. of the criminal(s). One cannot know the M.O. of the series without determining which crimes were involved, and one cannot locate the set of crimes without knowing the M.O.—the M.O. and the set of crimes need to be determined simultaneously. Pattern analysis for crime has existed at least as far back as the 1840s,[2] but recently, big data has changed the whole landscape for crime analysis. To locate crime patterns, analysts now rely on large amounts of very detailed data about large numbers of past crimes. The computational problem of finding crime series grows exponentially with the numbers of

[1]*Massachusetts Institute of Technology, Cambridge, Massachusetts.*
[2]*Cambridge Police Department, Cambridge, Massachusetts.*

*Address correspondence to: *Cynthia Rudin, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139,* E-mail: rudin@mit.edu

crimes and features of crimes. Even if a crime analyst could identify crime series within his/her own department database (which is already difficult), these manual efforts cannot scale, for instance, when neighboring police departments start to combine databases.

Despite its critical importance for public safety, there is little in the way of tools to help police. Most predictive policing software has the capability to detect only general background levels of crime density in time and space, which are much easier to predict than specific patterns of crime. This is called hotspot prediction, and it requires only time and location data, and a density estimation algorithm: Crimes are predicted to occur based on where and when they occurred in the past. No detailed information about the M.O. of the crimes is used for hotspot prediction, and hotspots can involve crimes committed by different offenders (as opposed to crime series, which have the same offenders). In fact, at least in the case of Cambridge, Massachusetts, crime series generally do not take place in hotspots (see also Ref.[3] for a study of crime series in two other U.S. cities). Hotspot prediction software is not what we need for the problem of crime series detection.

The problem of crime series detection is much more data intensive and computationally harder than hotspot prediction. To determine the M.O., we require very fine-grained detailed information about past crimes: where the offender(s) entered (front door, window, etc.); how they entered (pried doors, forced doors, unlocked doors, pushed in air conditioner, etc.), whether they ransacked the premise, the type of premise, etc. This is high dimensional structured data, leading to a computationally challenging data mining problem of finding all crimes that are similar to each other in several ways.

Crime series detection results can be used for many purposes. First, if a pattern is identified, investigative resources can be prioritized to focus on the crime series, to gather and assemble evidence. For instance, video recordings from nearby streets and stores or banks can be examined for evidence of the offender's location with respect to all crimes in the series, and combined with suspect descriptions from witnesses (if any). Call-detail records from cell phones can also be used for this purpose, as well as latent fingerprints and tracking information from stolen property. Second, if a current crime series is localized enough to have predictable times and locations (for instance, a pickpocket targeting a single café at regular intervals throughout the week), actions can be taken to stop the pattern. Third, if a current crime series has the same M.O. as a past crime series for which the offender is known, then the suspect from the older series could be a potential offender for the current crime series. Fourth, crime series detection results can be used to study criminal behavior generally. There is strong evidence that a majority of crimes are committed by a small number of serial criminals,[4] which underscores the importance of identifying and studying the patterns of serial offenders.

Conversely, we remark that nothing can be done if police do not know that a pattern is occurring at all. Without the capability of automatically detecting a specific series of crime, it is possible that crime series may take much longer to identify, or may never be identified. This is especially problematic for certain types of crime—for instance, for housebreaks (burglaries) there is often no suspect information since the crimes take place when residents are not present. Housebreaks can be extremely difficult crimes to solve, and nationwide only 14% of housebreaks are solved.[5] In this article, we aim directly at identifying series in housebreaks in an automated way.

The main hypothesis in this work is that most crime patterns have a *core* of crimes that exemplify the M.O. of the series. A core might be approximately three or four crimes that are very similar to each other in many different ways. According to our hypothesis that each crime series has a core, *if we can locate all small cores of crime within the dataset, we have thus located pieces of most crime series*. This hypothesis is based on the intuition of analysts and has the dual purpose of assisting with computation: We can indeed consider all reasonable small subsets to calculate whether they are plausible cores.

Though we cannot determine the M.O. of a series before we see it, we can characterize parts of the M.O.s we expect generally—for instance, crimes in a series are often close in time and space. We define a *pattern-general* similarity that encodes factors that are generally common to most crime series. It is learned from past crime series; for instance, proximity in time and space highly contribute to the pattern-general similarity. The pattern-general similarity induces a *similarity graph* over the set of crimes, where there is an edge between two crimes when they are similar according to the pattern-general similarity.

Crimes in a series also have *pattern-specific similarity*, where crimes are similar to each other if they all share the same M.O. The pattern-specific similarity and pattern-general similarity (the similarity graph in particular) are used for detecting cores: A core must

have both sufficiently large pattern-general similarity and pattern-specific similarity. The cores are found using an integer linear programming (ILP) formulation. Once the cores are found, we construct the full crime series by merging overlapping cores. We also prove that merging cores preserves desired properties.

Our method is a general subspace clustering method that can be used beyond crime series, and for other completely different application domains. It is novel in that it considers both pattern-general and pattern-specific aspects of patterns, where "pattern-general" means that it is common to many crime series (supervised from past clusters), and "pattern-specific" meaning aspects of a particular crime series (unsupervised). Our method does not force all examples to be part of a cluster and can thus accommodate background (non-series) crime. Our method also can find subspace clusters whose subspace morphs dynamically over the cluster; an M.O. can change when an offender becomes more sophisticated, adapts his M.O. to avoid detection, or when his preferred method is not available (e.g., he prefers entry through unlocked rear doors but will push in the air conditioner if his preferred means of entry is not available).

This project highlights several important aspects of big data analytics. It highlights an emphasis on human collaboration within the analysis pipeline, which is a key challenge for big data (see, for instance, the CCC white papers[6]). This project also exemplifies a separate challenge, namely that of increased complexity. Once our problem is formulated, solving it requires a combinatorially hard optimization problem and an explosion of variables. We could, for instance, add a "V" for "Variables" to the traditional V's of big data in order to capture problems that require massive computation to solve a problem of increased complexity. The big data challenge of problems with increased complexity has arisen in other places (see, for instance, the American Statistical Association's white paper on big data[7]).

Our three methodology sections follow the three main components of our method: *learn a similarity graph*, *mine cores*, and *merge cores*. We then show experiments where our method was tested on the full housebreak database from the Cambridge Police Department containing detailed information from thousands of crimes from over a decade. This method has been able to provide new insights into true patterns of crime committed in Cambridge. One observation that has been revealed in our experiments is that computers do not have the same biases that humans do. Computers search through the database in a different way than a human would, and thus can work symbiotically with analysts, showing them avenues to consider where they would not have normally ventured.

Related work

There are at least three main types of recent approaches to identifying crimes committed by the same individual or group (according to Ref.[8]).

The first approach, sometimes known as *pairwise case linkage*, involves identifying whether a pair of crimes was committed by the same group, where each pair is considered separately. Some of these works use weights determined by experts[9–12] to weight the various types of similarities between crimes, and other works learn the similarity weights from data.[13–23] The problem with considering only pairwise linkages is that only one similarity measure between crimes is used. This has a fundamental flaw that it does not consider M.O.'s of individual crime series. Consider two crime series with different M.O.'s: one has an M.O. in which the means of entry is to push in an air conditioner on a ground floor apartment (in buildings that are not concentrated geographically), and the M.O. of the other crime series involves breaking into a single family home at night (not using a consistent entry method) while the residents are present in a small geographical area. If we are investigating whether a new crime fits into the first series, logically we should consider whether the suspect entered by pushing in an air conditioner of an apartment and we should not consider geographical area. When we are investigating the second series, we should consider geographical area, time of day, and whether residents were present; we should not consider means or location of entry. If we use a single measure to judge similarity between any two crimes (as in pairwise case linkage approaches), it is not possible to make this distinction. In pairwise linkage approaches, it is not possible to ignore certain types of information about similarity and pay attention to others, as we claim is necessary for understanding multiple different M.O.'s. As a result, these approaches generally find that time and space are the only relevant dimensions for this type of analysis (see for instance Ref.[20]), and end up ignoring the detailed behavioral information.

The second type of approach is called *reactive linkage*, where a crime series is discovered one crime at a time, starting from a seed of one or more crimes (as defined by Refs.[1,8]). Reactive linkage has a similar problem to pairwise linkage in that when we start to grow a set

of crimes greedily from a small seed of one or two crimes, we cannot yet know the M.O. of the crime series. This means that yet again we start from a common similarity metric between crimes. The greediness of this approach, coupled with the fact that the distance metric does not take into account the M.O., could lead to problematic results. Our past approach to this[24,25] did use a theoretically motivated[26] greedy method, but adapted the distance metric to be closer to the observed M.O. as more crimes were added to the set. This still does not solve the problem of greediness, and the result could depend on which crimes were used as seed crimes. On the other hand, these greedy methods are very computationally tractable.

In the last type of approach, *crime series clustering*, all the clusters are found simultaneously.[8,27–35] One of the earliest approaches we know of for clustering crimes is that of Dahbur and Muscarello,[29] who used a neural network approach. (This method had some serious flaws that required extensive heuristic post-processing after the clusters were created, but aimed at solving the more general problem of crime clustering.) Most of these approaches use a form of hierarchical clustering, which again has the disadvantage that the distance metric between crimes is static and does not reflect the M.O. of any particular crime series. (On the other hand, hierarchical clustering is very computationally tractable and could be made to reflect pattern-general similarity, though not pattern-specific similarity.) Some of these approaches reduce features one at a time through hypothesis tests, or use basic dimensionality reduction (multidimensional scaling) techniques before clustering. This still does not handle pattern-specific aspects, and thus, cannot capture the M.O.

A main point of the present work is that in order to model the M.O., we need to use some form of subspace clustering. The M.O. for the pattern is precisely the subspace; it is the set of dimensions for which crimes in a particular series should be considered to be similar. For the first crime series in the example above, we would consider mainly the subspace "means of entry" involving the entry by pushing in air conditioners, and we would not heavily consider the dimension for geographical area. The algorithm must determine which observations go into which cluster, and which subspace is relevant for each cluster. Our work relates to various subfields of clustering (see, for instance, Ref.[36]), including pattern-based clustering,[37,38] which is a semi-supervised approach (unlike ours—we do not use test data at training time); general subspace clustering (e.g., Refs.[39,40]), which detects all clusters in all subspaces; and work at the intersection of dense subgraph mining and pattern mining in feature graphs.[41,42] These methods would not be able to take into account the complexities we handle, for instance, learning the pattern-general weights, or finding cores with the characteristics we require. Other work on space-time event detection is relevant,[43,44] where the goal is to detect patterns such that the frequency of records is higher than an expected frequency. A relevant method for clustering with simultaneous feature selection is that of Guan et al.,[45] which is similar to our core detector in flavor, although feature selection is controlled differently, and there are no pattern-general aspects.

Reviews of other data mining applications in crime analysis are those of Chen et al.[46] and Thongtae and Srisuk.[47] There are several other works that use machine learning for various criminology applications.[48–50]

Model Formulation

Our data consist of entities (crimes) $\mathcal{V}$, each of which has a vector of features. We define D_j as the set of possible values for feature j, $j = 1 \ldots J$. In our specific database we have features including time, space, whether the crime occurred on a weekday, location of entry (front door, back door, ground window, etc.), means of entry (shoved, pried, forced, etc.), type of premise (apartment, single-family house, etc.), indicator variable for "ransacked," suspect information (which is rarely present), and so on. Although the approach we introduce below is general and can be applied to problems beyond crime data mining, we use terminology specific to our problem in our exposition.

Define s_j to be a symmetric similarity function on the jth feature $s_j : D_j \times D_j \rightarrow [0, 1]$, where $s_j(v_l, v_k)$ measures the similarity between crimes v_ℓ and v_k in the jth feature. In our case most features are categorical, some are ranges, and some are spatial coordinates. For instance, if two crimes have location of entry as "window," then they would get a high location-of-entry similarity. The similarity measures are discussed in depth in our earlier work,[24,25] so we will not go into detail here. The average similarity of a set of crimes in feature j is defined as the cohesion of the set:

Definition 1. *(**Cohesion**$_j$) For a set of crimes V, the cohesion in the jth feature is the mean of pairwise similarities,*

$$Cohesion_j(V) = \frac{1}{|V|(|V|-1)} \sum_{v_l, v_k \in V} s_j(v_l, v_k).$$

The defining features of a pattern are those with sufficiently high cohesion.

Definition 2. *(**Defining feature**) Defining features for V are those that satisfy $Cohesion_j(V) \geq \theta_j$. The set of defining features for set V is denoted by $\Lambda(V)$.*

The defining features characterize the M.O. of the crime series. If several housebreaks happen in the same neighborhood, around the same time of day, within the same month, and the location of entry is always a window, regardless of the differences in other features, these similarities would indicate that the crimes could have been committed by the same offender. The features "geographic location," "time of day," "time between crimes," and "location of entry" characterize the M.O. for this particular crime series. When we later define cores, the pattern-specific statistic of interest is the number of defining features of V.

Let us switch from pattern-specific definitions to pattern-general definitions. We will learn a pattern-general similarity function from past crime series. Pattern-general similarity allows us to weight important features highly; for instance, crimes that are spread very far apart in time and space are unlikely to be a pattern, and thus we will learn from past crime series that time and space are important features. We will learn a set of weights $[\lambda_1, \ldots, \lambda_j, \ldots \lambda_J]$ from past crime data that will provide the importance of each feature within a linear combination. We will search for cores that have high pattern-general cohesion, defined as follows:

Definition 3. *(**Pattern-general cohesion**) The pattern-general cohesion of V is the weighted sum of cohesions over the features: $\sum_{j=1}^{J} \lambda_j \; Cohesion_j(V)$.*

Our cores will also be connected in the similarity graph. To construct edges for the graph, we first define a metric to measure the similarity between two crimes, as follows:

Definition 4. *(**Pattern-general similarity**) The pattern-general similarity is a weighted sum of similarity measures for each feature, using the pattern-general weights $[\lambda_1, \lambda_2, \ldots \lambda_J]$.*

$$\gamma(v_\ell, v_k) = \sum_{j=1}^{J} \lambda_j s_j(v_\ell, v_k). \tag{1}$$

We define the similarity graph to contain edges between crimes that are sufficiently close in the pattern-general sense.

Definition 5. *(**Similarity graph**) A similarity graph is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots v_n\}$ is the node set, $\mathcal{E}$ is the edge set, $\mathcal{E} = \{\{v_\ell, v_k\} | \gamma(v_\ell, v_k) \geq \Delta, v_\ell, v_k \in V, v_\ell \neq v_k\}$.*

For us to even consider whether V should be a core of a series, the core needs to be connected in the graph theoretic sense (and is not composed of two separate clusters for instance), must have sufficient connectivity, and must have sufficiently high pattern general cohesion. The requirement that a core must be a connected set means that each crime in a core must be similar in pattern-general similarity to at least one other crime in the core.

We learn both the weights $\{\lambda_j\}_j$ and cut-off threshold Δ from data, as discussed in Section 3 below, in order to perform the first piece of the method, which is to construct the similarity graph. Note that we could eliminate the first piece of the method, and eliminate the pattern-general similarity all together by setting the threshold Δ to be very low, and that way we consider only pattern-specific similarity; this, however, would not ease computation or restrict the cores to be similar to those from other patterns. Imposing a graph structure on the crimes eases computation, in the sense that we are now only looking for connected sets of the graph when we mine for cores in the second part of the method.

The second part of the method is to mine for cores. When we mine for cores, we cannot simply maximize the number of defining features and/or the pattern-general cohesion over all subsets of crime, as it would favor choosing very small sets of crime as cores. To alleviate this problem, we specify the size of the core $|V|$ and the number of defining features d, and find cores that maximize the pattern-general cohesion. Once we find the cores, we merge them to find the rest of the pattern. We call patterns formed by merging other patterns together *serieslike* patterns. An illustration of a serieslike pattern is shown in Figure 1.

Learning the Similarity Graph

As we discussed, the similarity graph is constructed by connecting pairs of nodes with pattern-general cohesion above a threshold Δ. The pattern-general cohesion γ is defined in (1) as a weighted sum of pairwise similarities in different features, with pattern-general weights $\lambda \in \mathcal{R}^J$. The set of coefficients λ and Δ are parameters that we learn from data. These data consist of 51 historical crime series that have been identified as true crime series by crime analysts.

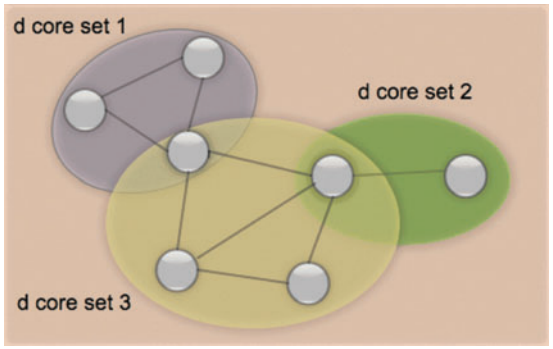

FIG. 1. A *d* serieslike pattern consisting of three *d* cores of various sizes.

To learn the weights, we optimize over the historical training patterns to make them as close as possible to being connected subgraphs. This means we want each crime in a historical pattern to be close to at least one other crime in the same pattern. At the same time, we want crimes within a historical pattern to be distant from crimes not in the same pattern. The condition for crime ℓ to be close to at least one other crime within its pattern (with some slack ϵ_ℓ) is:

$$\max_{\{k:k\in\text{pattern}(\ell)\}} \sum_{j=1}^{J} \lambda_j s_j(\ell,k) \geq \Delta - \epsilon_\ell. \tag{2}$$

Conversely, crimes that do not belong to the same pattern should not be very similar:

$$\sum_{j=1}^{J} \lambda_j s_j(\ell,k) \leq \Delta + \xi_{\ell k}, \tag{3}$$

true for all ℓ and k such that $k \notin$ pattern(ℓ). Our goal is to minimize the total weighted slack. Informally, we compute:

$$\min_{\lambda,\Delta} \sum_{\substack{\text{crimes in the}\\ \text{same pattern}}} \text{slack} + C_1 \sum_{\substack{\text{crimes not in the}\\ \text{same pattern}}} \text{slack} + C_0 \|\lambda\|_0,$$

where $\|\lambda\|_0$ is the ℓ_0 semi-norm of λ, which encourages sparsity in λ. The constant C_1 is set very small, so most edges that should be present will be present, and we consider removal of unnecessary edges as a secondary goal. Removal of these unnecessary edges is where we will gain a computational benefit later on. The more edges we eliminate, the fewer connected subsets we need to evaluate as being possible cores.

We propose a mixed-integer programming (MIP) formulation for solving this. In what follows, decision variables $y_{\ell k}$ are binary, and they select a crime k that is most similar to a given crime ℓ within the same pattern. That is, they encode the max from constraint (2). The formulation is:

$$\min_{\lambda,\Delta,\{y_{\ell,k}\}_{\ell,k},\{\beta_j\}_j} \sum_{\ell\in\{1,2\ldots m\}} \epsilon_\ell + C_1 \sum_{\substack{\ell\in\{1,2\ldots m\}\\ \{k:k\notin\text{pattern}(\ell)\}}} \xi_{\ell k} + C_0 \sum_{j=1}^{J} \beta_j$$

such that

$$\sum_{j=1}^{J} \lambda_j s_j(\ell,k) \geq (\Delta - \epsilon_\ell) + M(y_{\ell k} - 1) \;\; \forall \ell \;\; \forall k \text{ s.t. } k \in \text{pattern}(\ell) \tag{4}$$

$$\sum_{k:k\in\text{pattern}(\ell)} y_{\ell k} = 1 \qquad \forall \ell \tag{5}$$

$$y_{\ell k} \in \{0,1\} \qquad \forall \ell, k \tag{6}$$

$$\sum_{j=1}^{J} \lambda_j s_j(\ell,k) \leq \Delta + \xi_{\ell k} \qquad \forall \ell, \forall k \text{ s.t. } k \notin \text{pattern}(\ell) \tag{7}$$

$$\sum_{j=1}^{J} \lambda_j = 1 \tag{8}$$

$$\lambda_j \geq 0 \qquad \forall j \tag{9}$$

$$\lambda_j \leq \beta_j \qquad \forall j \tag{10}$$

$$\beta_j \in \{0,1\} \qquad \forall j. \tag{11}$$

Constraint (4) comes from (2). It forces $y_{\ell,k}$ to be chosen correctly so that if k is the crime closest to ℓ, then $y_{\ell,k}$ will be 1. This is because ϵ_ℓ is minimized within the objective, so $y_{\ell,k}$ is necessarily going to correspond to the index where ϵ_ℓ is minimized, and where the similarity is maximized within (4). Constraints (5) and (6) further define $y_{\ell,k}$'s by stating that a crime in the pattern needs only to be connected to one closest neighbor ℓ in the pattern (this is the requirement of connectivity), and its entries are binary. Constraint (7) comes from (3). The value $\sum_j \beta_j$ is the ℓ_0 norm of λ. The β_j's are decision variables where if $\beta_j = 1$, λ_j is nonzero. This formulation is linear, and thus using MIP technology there is a guarantee on the optimality of the solution. In particular, the solver will provide the duality gap, and when it is zero, we know that the optimal solution to the optimization problem has been attained.

Cores

A *d-core* is a set of crimes that exhibit similarity in a feature subspace of d dimensions. A d-core has d defining features that are not predetermined. Further, crimes in a d-core need to be well connected in the similarity graph. Formally, the definition is as follows.

Definition 6. *(**d-core**) A similarity graph $G=(V, E)$ with density threshold α is called a d-core if it satisfies the core constraints:*

- *Pattern specific constraint: the size of the defining feature set of the graph is equal to d, $|\Lambda(G)|=d$.*
- *Pattern general constraints: G is connected, and G is dense, $\frac{|E|}{|V|(|V|-1)} \geq \alpha$. That is, the fraction of possible edges in the graph exceeds α.*

Two parameters, d and α, control the property of the core in a pattern-specific and pattern-general way, respectively.

The pattern-general constraints should be thought of as being much looser than the pattern-specific constraints, as we often include unnecessary edges in the similarity graph. For the set of crimes to be feasible in the pattern-specific sense is much more difficult, as crimes in the pattern need to be similar to each other in d separate ways.

We find cores $G=(V, E)$ using an optimization method to maximize the pattern-general cohesion while satisfying the core constraints.

$$\begin{aligned} &\underset{G}{\text{maximize}} && \sum_{j\in J} \lambda_j \text{Cohesion}_j(G) \\ &\text{subject to} && |V|=n \\ &\text{core constraints}: && \begin{cases} |\Lambda(G)|=d, \\ G \text{ is connected}, \\ \frac{|E|}{|V|(|V|-1)} \geq \alpha. \end{cases} \end{aligned} \tag{12}$$

We propose a binary integer linear formulation for the optimization problem (12). Let $m=|\mathcal{V}|$, the number of crimes in the database. Let n be the size of the pattern we want to discover, and we loop through possible values of n, re-solving each time. We define an $m\times m$ similarity matrix for each feature $\mathbf{S}_1, \ldots, \mathbf{S}_J$ with elements $\mathbf{S}_j(\ell, k)=s_j(v_\ell, v_k)$, which are precomputed. Let $\mathbf{X}$ be the matrix of binary decision variables defining the core. $\mathbf{X}(\ell, k)$ is 1 if a pair of crimes ℓ and k are in core G, which is the same as $\mathbf{X}(k, \ell)$, so matrix $\mathbf{X}$ is symmetric. On the diagonal, $\mathbf{X}(\ell, \ell)$ represents whether crime ℓ is in core G. We use $d_1, \ldots, d_J \in \{0, 1\}$ to indicate whether feature j is a defining feature, or equivalently, whether $\text{Cohesion}_j(G) \geq \theta_j$. Let $\mathbf{E}$ be the adjacency matrix for the (pattern-general) similarity graph, where $\mathbf{E}(\ell, k)=1$ if $\{v_\ell, v_k\} \in E$, and $\mathbf{E}(\ell, k)=0$ otherwise. Since the graph is undirected, $\mathbf{E}$ is symmetric. We set $\mathbf{E}(\ell, \ell)=1$ for computational simplicity. $(\mathbf{E}\circ\mathbf{X})$ is the Hadamard product of matrix $\mathbf{E}$ and $\mathbf{X}$, where $(\mathbf{E}\circ\mathbf{X})(\ell, k)=\mathbf{E}(\ell, k) \cdot \mathbf{X}(\ell, k)$. M is a large auxiliary parameter for formulating the problem with a big-M formulation, and ϵ is small. With this notation, the optimization problem (12) can be reformulated:

$$\max_{\mathbf{X}, \{\mathbf{d_j}\}_\mathbf{j}} \quad \sum_j \lambda_j \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) \text{ s.t.}$$

$$\sum_\ell \mathbf{X}(\ell, \ell)=n \tag{13}$$

$$\frac{1}{n(n-1)} \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) - Md_j \leq \theta_j - \epsilon \qquad \forall_j \tag{14}$$

$$\frac{1}{n(n-1)} \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) - Md_j \geq \theta_j - M \qquad \forall_j \tag{15}$$

$$\sum_{j\in J} d_j = d \tag{16}$$

$$\mathbf{X}(\ell, k)=\mathbf{X}(k, \ell) \quad \forall \ell, k \tag{17}$$

$$\mathbf{X}(\ell, k) \leq \mathbf{X}(k, k) \quad \forall \ell, k \tag{18}$$

$$\mathbf{X}(\ell, \ell) + \mathbf{X}(k, k) \leq \mathbf{X}(\ell, k) + 1 \forall \ell, k \tag{19}$$

$$(\hat{\mathbf{E}}^{n-1} \circ \mathbf{X})(\ell, k) \geq \mathbf{X}(\ell, k) \quad \forall \ell, k \tag{20}$$

$$\mathbf{X}(\ell, k), d_j \in \{0, 1\} \quad \forall \ell, k, j \tag{21}$$

where matrix $\hat{\mathbf{E}}$ is defined just below.

Let us derive the objective. Since $\mathbf{X}(\ell, k)=1$ if and only if both crimes ℓ and k are in the core (that is, they are in graph G that we discover), we have the following:

$$\frac{1}{n(n-1)} \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) = \text{Cohesion}_j(G). \tag{22}$$

The objective is the pattern-general cohesion. Equation (13) ensures that the cores we discover are of size n. Constraints (14), (15), and (16) are the pattern-specific constraints. Constraint (14) forces $d_j=1$ when $\text{Cohesion}_j(G) \geq \theta_j$, where the strict inequality is enforced by ϵ. Constraint (15) forces $d_j=0$ when $\text{Cohesion}_j(G) < \theta_j$. Constraint (16) specifies the number of defining features as d. The symmetry of $\mathbf{X}$ is enforced by (17).

Constraints (18) and (19) imply $\mathbf{X}(\ell, k)=1$ iff both $\mathbf{X}(\ell, \ell)$ and $\mathbf{X}(k, k)$ are 1 and 0 otherwise.

Expression (20) is a pattern-general constraint. Our formulation does not enforce the core to be connected, but it does enforce something weaker, namely that each node in a core of size n is at most $n-1$ steps along the similarity graph from any other node in the core. We handle the connectivity afterward by examining the result to ensure that it is connected and labeling it as infeasible if not. To handle the constraint that each node in the core is at most distance $n-1$ from every other node, we recall that in graph theory, if node v_k is reachable from node v_ℓ in exactly q steps, $\mathbf{E}^q(k, \ell)>0$. If node v_k is reachable from node v_ℓ in at most $n-1$ steps, it means that at least one of $\mathbf{E}^q(k, \ell)>0$ for $q \leq n-1$. We define the following matrix $\hat{\mathbf{E}}^{n-1}$, where an element $\hat{\mathbf{E}}^{n-1}(k, \ell)$ indicates if node k and node ℓ are distance at most $n-1$ steps along the graph.

$$\hat{\mathbf{E}}^{n-1}(\ell, k)=\begin{cases} 1 & (\mathbf{E}+\mathbf{E}^2+\cdots+\mathbf{E}^{n-1})(\ell, k) > 0 \\ 0 & \textit{otherwise.} \end{cases}$$

Thus, (20) forces that if v_ℓ and v_k are both in the pattern, they must be at most distance $n-1$ along the graph.

The pattern-general density constraint is handled similarly to the connectivity constraint, where feasibility is checked for each solution, and infeasible solutions are removed.

The integer program finds one solution at a time. In order to avoid finding a solution that was found previously, we introduce a constraint for each previous solution found. Suppose in the t-th run, the crimes in the solution are $\mathcal{Q}_t$, that is, $\mathbf{X}(k, k)=1$ if $k \in \mathcal{Q}_t$, $\mathbf{X}(k, k)=0$ otherwise. The constraint we add before running the $t+1$-th time is

$$\sum_{k \in \mathcal{Q}_t} \mathbf{X}(k, k) \leq n-1. \tag{23}$$

This constraint will exclude the current solution from the feasible region, and we will obtain a different solution in the next run if any feasible solutions remain. Since all of the matrices are symmetric, in practice we keep only the upper (or lower) triangle, including the diagonal, to compute all of the sums.

Merging Cores

By our main hypothesis, the vast majority of crime series contain a core. This means that by finding all cores, we would have located the vast majority of all crime series. We now grow the rest of the crime series from the cores by merging them together. One advantage of merging is that it allows the pattern to dynamically change, as the defining features from the merged cores are not always equal to each other. Consider a burglar's M.O. with a shifting means of entry. At first he enters through unlocked doors, then he starts to use bodily force to open doors, and later he learns to use a screwdriver to pry the door open. His full pattern thus consists of several smaller d-cores. This suggests a more flexible definition of pattern than a simple core. We provide such a definition below.

Definition 7. *(**d-serieslike pattern**) A graph $G=(V, E)$ is called a d-serieslike pattern with defining feature set $\Pi(G)$ of size d if it satisfies:*

- *Pattern general constraint: G is connected.*
- *Pattern specific constraint: Each node u in G is contained in at least one subgraph $G' \subseteq G$ that is a d'-core with defining features that include set $\Pi(G)$, that is, $\Pi(G) \subseteq \Lambda(G')$, $d \leq d'$.*

Note that if a graph is a d-serieslike pattern, it is also a $(d-1)$-serieslike pattern, a $(d-2)$-serieslike pattern, and so on. The pattern specific constraints for d-serieslike patterns are looser than those for d-cores. A d-serieslike pattern may not be a d-core. On the other hand, a d-core is a special case of a d-serieslike pattern. Before we proceed, we must ensure that merging is justified.

Theorem 1. *(**The set of serieslike patterns is closed under merging.**) Suppose G_1 is a d_1-serieslike pattern and G_2 is a d_2-serieslike pattern. If $G_1 \cap G_2 \neq \emptyset$, then $\hat{G}=G_1 \cup G_2$ is a d-serieslike pattern, with defining features $\Pi(G_1) \cap \Pi(G_2)$, $d=|\Pi(G_1) \cap \Pi(G_2)|$.*

Proof. First, since G_1 and G_2 are connected and $G_1 \cap G_2 \neq \emptyset$, the union of them is also connected, that is, $\hat{G}$ satisfies the pattern general constraint. Then for all nodes $u \in G_1$, $\exists$ a d_1-core G_u^1 such that $u \in G_u^1$ and $\Pi(G_1) \cap \Pi(G_2) \subseteq \Pi(G_1) \subseteq \Lambda(G_u^1)$, and for all nodes $u \in G_2$, $\exists$ a d_2-core G_u^2 such that $u \in G_u^2$ and $\Pi(G_1) \cap \Pi(G_2) \subseteq \Pi(G_2) \subseteq \Lambda(G_u^2)$. So, either way, the defining feature set includes $\Pi(G_1) \cap \Pi(G_2)$. This means that for any node $\hat{u} \in \hat{G}$, $\exists$ a core $\hat{G}_u$ such that $\Pi(G_1) \cap \Pi(G_2) \subseteq \Lambda(\hat{G}_u)$. ■

This leads directly to the following:

Corollary 1. *Suppose G_1 is a d_1-serieslike pattern, G_2 is a d_2-serieslike pattern, …, G_n is a d_n-serieslike pattern, and $G_1 \cup \ldots \cup G_n$ is connected,*

$$|\Pi(G_1) \cap \Pi(G_2) \cap \ldots \cap \Pi(G_u)|=d.$$

Then $G_1 \cup \ldots \cup G_n$ is a d-serieslike pattern.

These properties lead to the following breadth first search algorithm for mining *d*-serieslike patterns. We start with the cores that we found using the integer program. These cores are *candidates* for merging. We also maintain an *active pattern set* that contains the *d*-serieslike patterns that we are not done constructing. We keep the *d*-serieslike patterns that we are done constructing in a *maximal pattern set*. To start, the active pattern set contains all of the *d*-cores we found. For each active pattern, we iterate through the candidates to see if they meet the merging criteria provided just below. If a merge is possible, and if the merged set had not been previously created, we append the merged pattern to the active pattern set and continue iterating through the candidates. If there are no candidates that can be merged with the active pattern at all, then the active pattern is maximal, and it is placed in the maximal pattern list. The *merging criteria* for $G_1 \cup G_2$ to form a *d*-serieslike pattern $\hat{G}$ is

- $G_1 \cap G_2 \neq \emptyset$
- $|\Pi(G_1) \cap \Pi(G_1)| \geq d$.

The merging algorithm is formulated in Algorithm 1.

Algorithm 1: Merging Cores

```
INPUT: d, cores, each with ≥ d defining features
candidate list ← cores
active set ← cores, each with defining features
maximal set ← ∅
while active set ≠ ∅ do
  G_current ← any element in active set
  Π_current ← defining features from G_current
  isMaximal ← TRUE;
  for all G_j ∈ candidates, G_j ⊄ G_current do
    if G_current ∩ G_j ≠ ∅, |Π(G_current) ∩ Λ(G_j)| ≥ d then
      Ĝ ← G_current ∪ G_j
      Π(Ĝ) ← Π(G_current) ∩ Λ(G_j)
      if Ĝ does not exist in active set or maximal set then
        isMaximal ← FALSE;
        append Ĝ to active set
      end if
    end if
  end for
  if isMaximal = = TRUE then
    remove G_current from active set, put into maximal set
  end if
end while
OUTPUT: maximal set
```

Experiments

Our data set was provided by the Crime Analysis Unit of the Cambridge Police Department in Massachusetts. It has 7,067 housebreaks that happened in Cambridge between 1997 and 2011, containing 51 hand-curated patterns contained within the 4,864 crimes between 1997 and 2006. (Patterns from 2007 to 2012 were not assembled at the time of writing.) Crime attributes include geographic location, date, day of week, time frame, location of entry, means of entry, an indicator for "ransacked," type of premise, an indicator for whether residents were present, and suspect and victim information. The 51 crime series identified by police contain an average of 12.1 crimes each, with the largest series containing 59 crimes, and the smallest series containing 2 crimes. These crimes span an average period of 42 days, with the shortest series taking place within 1 day and the longest series taking 451 days. Data were processed using the similarity functions $\{s_j\}_j$ discussed in our previous work,[24,25] where each pairwise feature is mapped into a number between 0 and 1. These similarity measures are *p*-values, and they consider the baseline frequency of each possible outcome for the categorical variables. For instance, most crimes are committed when residents are not present. If two crimes were committed where one had residents present and the other had residents that were not present, then the similarity score for "residents present" is zero. If both crimes were committed when the residents were not present, the similarity score for "residents present" would not be high (in particular, it is $1 - p^2_{\text{not present}}$ where $p_{\text{not present}}$ is the proportion of crimes in the database where residents were not present), whereas if two crimes were committed with residents present, the similarity would be much higher ($1 - p^2_{\text{present}}$). The similarity score for time frames is complicated because it takes into account the distribution of times when crimes are more frequently committed. We took the 51 hand-curated patterns and divided them randomly into four subsets (folds) with sizes 12 or 13 patterns each. We used three of the four folds to learn the pattern-general weights and tested on the remaining fold for the experiments discussed below.

Baselines

As this problem is fundamentally a clustering problem, we compare with several varieties of hierarchical agglomerative clustering and incremental nearest neighbor approaches. For these baselines, we use several different schemes to iteratively add discovered crimes, starting from pairs of nodes with high similarity γ, which is a weighted sum of the attribute similarities:

$$\gamma(C_i, C_k) = \sum_{j=1}^{J} \hat{\lambda}_j s_j(C_i, C_k).$$

Unlike our method where the weights are learned, the weights $\hat{\lambda}$ for the baselines were provided by crime analysts based on domain expertise, similar to several other works.[10,11]

Hierarchical agglomerative clustering (HAC) begins with each crime as a singleton cluster, and iteratively merges the clusters based on the similarity measure between clusters. *Nearest neighbor classification* (NN) first selects pairs of crimes with high similarity and then iteratively grows a cluster by adding the nearest neighbor crime to the cluster.

HAC and NN were used with three different criteria for cluster–cluster or cluster–crime similarity: *single linkage* (SL), which considers the most similar pair of crimes; *complete linkage* (CL), which considers the most dissimilar pair of crimes; and *group average* (GA), which uses the averaged pairwise similarity.[51] When the nearest neighbor algorithm is used with the S_{GA} measure defined below with weights provided by crime analysts, it is similar to the Bayesian Sets algorithm and how it is used for set expansion.[52,53]

$$\begin{aligned} S_{SL}(G_1, G_2) &:= \max_{v_k \in G_1,\, v_\ell \in G_2} \gamma(v_k, v_\ell) \\ S_{CL}(G_1, G_2) &:= \min_{v_k \in G_1,\, v_\ell \in G_2} \gamma(v_k, v_\ell) \\ S_{GA}(G_1, G_2) &:= \frac{1}{|G_1||G_2|} \sum_{v_k \in G_1} \sum_{v_\ell \in G_2} \gamma(v_k, v_\ell). \end{aligned} \tag{24}$$

Evaluation metrics

There are two levels of performance we evaluate - *pattern-level* and *object-level*.

Pattern-level precision and recall

We evaluate the quality of the core detector using *pattern-level* precision and recall. The *d*-cores are smaller as *d* becomes larger. The cores are used for discovering larger merged patterns. Thus we evaluate the accuracy of the core finder in its detection ability; if a real pattern is missed completely by our core detector, there is no way to recover from this in order to detect it. If a core covers more than one pattern, this is also a bad seed for further mining, since it would generate misleading defining features that do not characterize any real patterns. Thus, we call cores that cover one and only one real pattern *good* cores. The pattern-level precision and recall are both defined using good cores. N denotes the number of cores we discover.

$$\text{P-Precision (cores)} = \frac{\sum_i^N \mathbf{1}(\text{core } i \text{ is } \textit{good})}{N} \tag{25}$$

$$\text{P-Recall (cores)} = \frac{\sum_i^N \mathbf{1}(\text{core } i \text{ is } \textit{good})}{|\mathcal{P}|}. \tag{26}$$

Note that pattern-level precision should be large, as each real pattern should contain many cores, inflating the reported precision values.

Object-level precision and recall

We evaluate the full pipeline for generating serieslike patterns using *object-level* precision and recall. To do this, for each pattern discovered, we determine how close it is to one of the real patterns. If the discovered pattern overlaps only one real pattern, then we call this the *dominating pattern* and evaluate precision and recall with respect to crimes in that pattern. If the serieslike pattern overlaps more than one real pattern, we assign the dominating pattern to be the real pattern possessing the most crimes that overlap with our discovered pattern. Note that it is possible for the recall not to grow with the size of the discovered pattern, as the dominating real pattern could change as the discovered pattern grows larger. The definitions of object-level precision and recall for a *d*-serieslike pattern $G = (V, E)$ are as follows:

$$\text{O-Precision(G)} = \frac{\sum_{\ell=1}^{|V|} \mathbf{1}(\ell \in \text{dominating pattern})}{|V|} \tag{27}$$

$$\text{O-Recall(G)} = \frac{\sum_{\ell=1}^{|V|} \mathbf{1}(\ell \in \text{dominating pattern})}{|V_{\text{dominating pattern}}|} \tag{28}$$

where $|V_{\text{dominating pattern}}|$ is the number of crimes in the dominating real pattern.

Computational gain from similarity graph

The first step in our method is to learn the similarity graph. The similarity graph provides a computational gain in that it creates constraints on possible cores, reducing the feasibility region of the ILP. For this similarity graph, recall that we desire crimes in the same real pattern to be connected to each other. We call edges connecting crimes that belong to the same pattern *good edges*. If we have constructed the similarity graph well, the similarity graph should have a higher percentage of good edges than if we had simply used the full graph consisting of all possible edges. If we remove a few good edges in the process, this is not problematic

as long as the true patterns are still connected in the similarity graph—this will be assessed when we assess the quality of the cores and the full pipeline next.

Table 1 shows the percentages of good edges in both the similarity graph and the full graph for four test folds (the data were divided into four folds, and each was used in turn as the test fold). The learning method tends to reduce the number of unnecessary edges by a factor of 7 or 8 in each of the test folds, as shown in the third column (which is the first column divided by the second column). This reduction substantially reduces computation for the core finder.

Pattern-general weights

The pattern-general weights come from the learning step for the similarity graph. In Figure 2 we report the mean over the test folds of the pattern-general weights we discovered. The highest weights are similarity in distance, number of days apart, suspect information, whether residents are present, and means of entry (e.g., pried, forced, cut screen).

Time windowing

Consider solving the ILP for finding cores on data from 4,864 housebreaks. Note that if we were to search for patterns of size 10 among 1,000 crimes, this would mean investigating $\binom{1000}{10} \approx 2.634 \times 10^{23}$ possible subsets. Further, CPLEX would need to handle $1{,}000^2$ constraints (18) and (19) of the optimization problem. Luckily it is unlikely that a crime series would possess a core of size 10, but still we need to find ways to reduce computation.

Because the pattern-general weight on closeness in time is so high, we determined that we would be unlikely to miss true cores if we considered windows of time that include at least 200 crimes. We thus indexed the housebreak records in chronological order and created overlapping windowed blocks of 200 crimes each, where neighboring blocks have an overlap of 100 crimes. Therefore we solve ILPs among crime subsets $\{1, \cdots, 200\}$, $\{100, \cdots, 300\}, \cdots, \{4700, \cdots, 4864\}$. In each subset, we input the number of defining features d and core size n, and then iteratively run the ILP to get all feasible solutions by adding the constraint (23) after each iteration to avoid returning repeated solutions.

Table 1. Test Results of Weights Learning Algorithm

	Good edges % in similarity graphs	*Good edges % in complete graphs*	*Reduction factor*	*Training time*
1	1.84	0.26	7.1	3.64×10^3 s
2	2.42	0.33	7.3	9.58×10^3 s
3	3.34	0.42	8.0	6.20×10^3 s
4	2.95	0.34	8.7	5.41×10^3 s

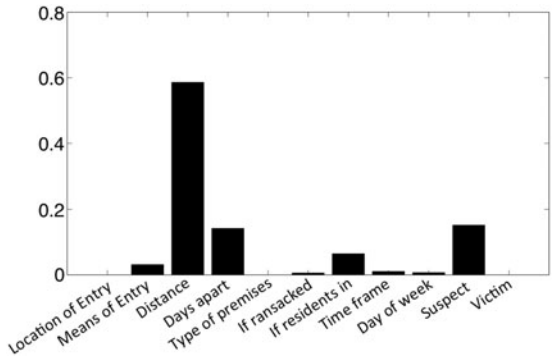

FIG. 2. Pattern-general weights.

Evaluation of mining cores

We chose performance evaluation metrics from information retrieval, and for some of these metrics, we need to rank the discovered cores by a scoring function. This scoring function represents how certain we are that these cores are real. We use a scoring function that is a weighted version of pattern-general cohesion and the (pattern-specific) number of defining features, as we desire cores that are both tight in the pattern-general sense and in the pattern-specific sense. Here is the score function:

$$\text{Score}(G) = \sum_{j=1}^{J} \lambda_j \text{Cohesion}_j(G) + \frac{1}{6} \cdot d. \qquad (29)$$

Series usually have about six defining features, so the choice of 1/6 tends to balance the two terms, weighing the pattern-specific term slightly higher. We ordered the discovered cores in decreasing order of the scores. Note that the first evaluation below does not require these scores, but the second and third do.

1) **Cores with different *d*** We expect that discovered patterns with more defining features are more likely to be true crime series. Figure 3 shows how the precision increases with the number of defining features d. In this figure, we consider only cores of the same size (three crimes). There are no overlapping cores between the three bars, as each core is used once with its exact number of defining features d, which is 6,

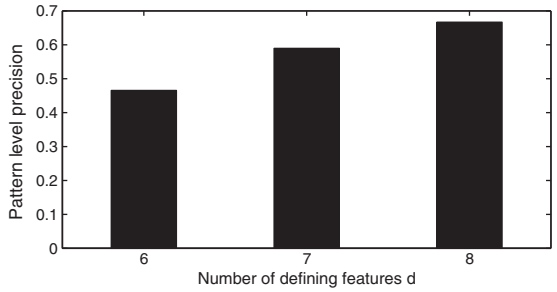

FIG. 3. Pattern-level average precision of d cores of different sizes.

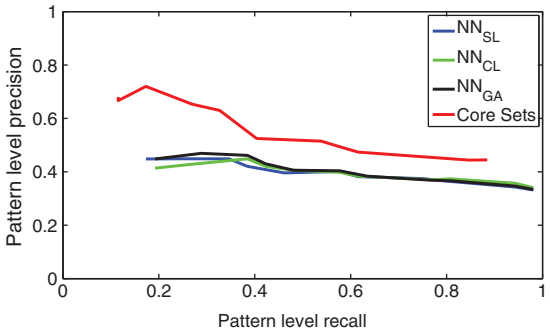

FIG. 5. Average pattern-level precision vs. recall.

7, or 8. The number of discovered cores for $d=6$ is 1072, $d=7$ is 215, and for $d=8$ is 36. There were too few cores with more than eight defining features to reliably calculate precision. The reported numbers of cores are totals from all test folds.

2) **Cores with different size *n*** We used the scoring function (29) as a filter to pick the best 1,000 cores, from each of the sizes 3, 4, and 5, and discarded the other discovered cores. Larger cores have higher chances of hitting a pattern, since there are more crimes in the core; however, they also have higher chances of hitting more than one real pattern. As shown in Figure 4, cores of size 4 have a much higher pattern-level precision than cores of size 3, but cores of size 5 do not have noticeable gains over size 4. This is because the increased probability of hitting a real pattern cancels with the increased probability of hitting more than one real pattern.
3) **P-precision–p-recall curve** We generated a full list of cores of size 3 with d between 6 and 8, and ranked the cores according to their scores. As we moved down the list, we evaluated pattern-level precision and recall at each step. We also did the same procedure with the baseline iterative nearest neighbor method used for generating cores of size 3, using all of the similarity measures in (24). (Note that for HAC we cannot control the size of cores for evaluation.) The precision-recall curves for all four methods averaged over the test folds are plotted in Figure 5. It is clear that our core finder is substantially better than the baselines, though that is not surprising given that it searches globally for the best cores.

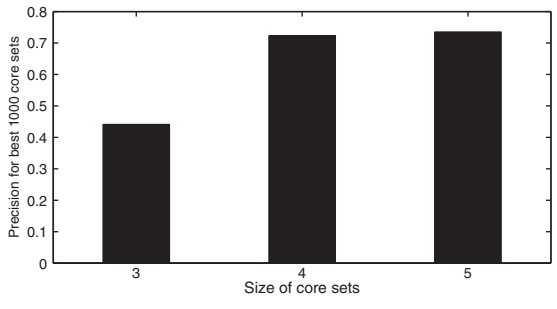

FIG. 4. Pattern-level precision of cores with different sizes.

Evaluation for mining serieslike patterns

We evaluated the quality of our full pipeline and the baseline methods as follows. After all the serieslike patterns were discovered, we evaluated the average object-level precision and recall for all the patterns and over all the test folds, plotted as a point on Figure 6. For HAC, we simply iterated it, stopping at a threshold where recall was approximately equivalent to our method, and again reported average object-level precision and recall on Figure 6. For the nearest neighbor method, after each element was added to a growing pattern, we evaluated precision and recall to trace out a precision-recall curve. All three metrics in (24) were used for HAC and nearest neighbors. We note that for the same level of recall, the precision attained by our method was quite a bit higher than that of other methods.

There is still a lot of room for improvement. Currently, with precision on the order of 53%, we capture approximately 18% of the crimes identified by analysts. That is, when our method returns a crime, it is a real crime in a series about 53% of the time. We are returning about one-fifth of the crimes at these settings, so our

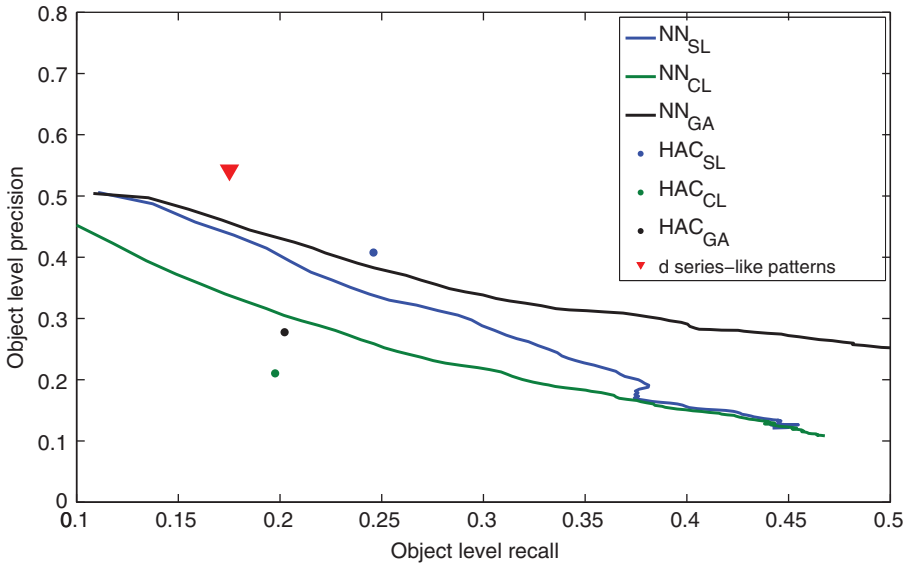

FIG. 6. Object-level precision and recall.

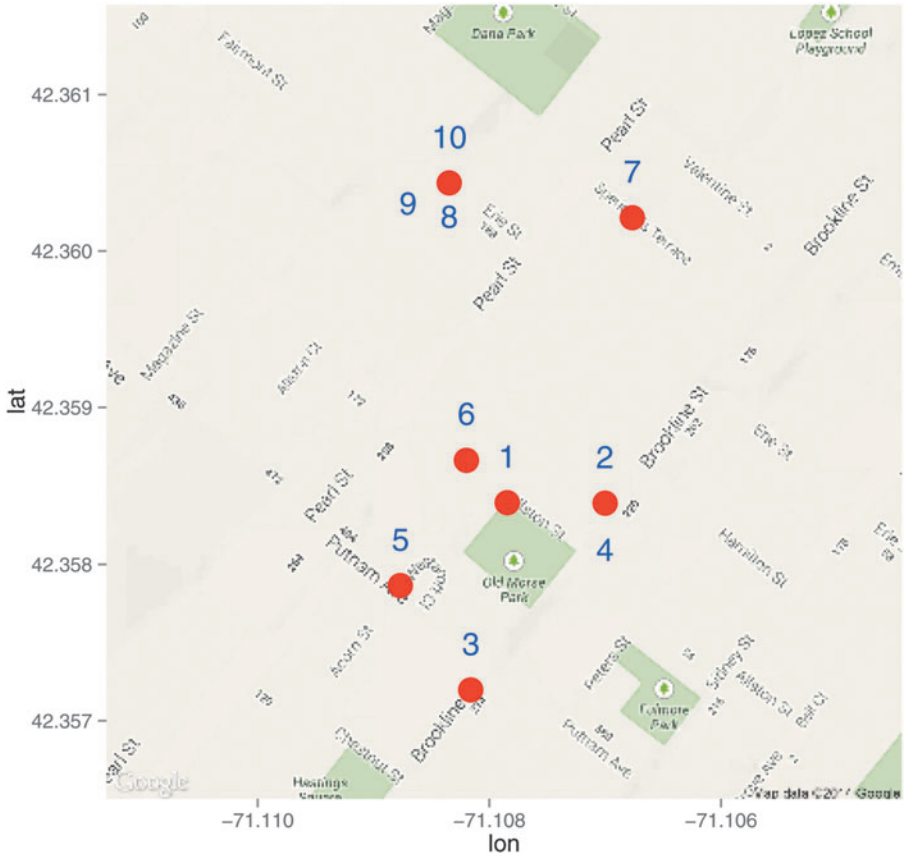

FIG. 7. The locations of crimes in the first series.

Table 2. Example 1 of a Serieslike Pattern with $d = 6$

No.	Date	Location of entry	Means of entry	Premises	Ransacked	Residents	Time of day	Day	Suspect	Victim
1	11/8/06	Basement door	Unknown	Unknown	No	Not in	10:45–15:00	Wed	Null	1 F
2	11/8/06	Front door	Pried	Unknown	No	Not in	8:00–18:30	Wed	Null	1 M
3	11/16/06	Front door	Shoved/forced	Unknown	No	Not in	9:00–17:00	Thur	Null	1 M
4	12/7/06	Front door	Pried	Unknown	No	Not in	9:00–17:00	Thur	Null	1 F
5	12/22/06	Front door	Pried	Unknown	No	In	11:48	Fri	Null	1 M
6	2/1/07	Front door	Shoved/forced	Unknown	No	In	14:45	Thur	3 Males	1 F
7	2/15/07	Front door	Unknown	Aptment	No	In	12:00–13:30	Thur	Null	2 F
8	3/5/07	Front door	Shoved/forced	Aptment	No	Not in	12:22–14:56	Mon	Null	White M
9	3/5/07	Front door	Broke	Aptment	No	Not in	12:22–14:56	Mon	Null	1 F & 1 M
10	3/8/07	Front door	Pried	Aptment	No	Not in	12:50–13:30	Thur	Null	1 M

F, female; M, male.

method currently is conservative–it will not claim that a crime is in a series unless it is reasonably certain. This can help ensure that analysts do not overlook crimes that are very likely to be in a particular series. Note that our ground truth consists of crime series that are hand-labeled by analysts, and these labels are not perfect. For instance, it is entirely possible that we return crimes that are in a series that the police had not previously considered. We might opt to change the settings so that the method returns more crimes as being potentially part of the series (giving lower precision but higher recall). This might be changed by making the core finder less conservative, finding a way to incorporate crimes that are not in cores, or possibly by working with domain experts to improve the database used for evaluation. We note that the baseline methods work surprisingly well and are themselves reasonable options.

Case Studies

We performed a blind test, where we aimed to detect crime patterns between 2007 to 2012, for which we do not have pattern data. The results were analyzed by hand by crime analysts.

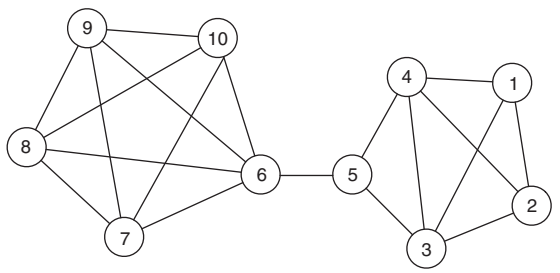

FIG. 8. Similarity graph for the first crime series.

Case study 1

One particularly interesting crime series includes 10 crimes from November 2006 to March 2007. Figure 7 shows geographically where these crimes were located. Table 2 provides some of the details about the crimes within the series.

From the (pattern-general) similarity graph, we isolated the subgraph containing the 10 crimes, which is diagrammed in Figure 8. Crimes 1 to 5 are well connected as a subset, and crimes 6 to 10 are well connected as another subset. From only the similarity graph, the two subsets do not seem very related except for a single edge between crimes {5, 6} connecting them; however, this is only the pattern-general part of the story.

We used the integer linear program (12) to discover cores of size 3 with at least 6 defining features. Table 3 lists the cores and their defining features in this series, where a check mark means the feature is a defining feature and a circle means it is not. These cores show how the crimes are similar to each other in a pattern-specific way. The next step is merging the cores. The defining feature set $\Pi(G)$ was chosen to include six features, which are geographic location, days apart, location of entry, the ransacked indicator, time of day, and day of the week. One of the cores, core 16 in Table 3, which was not included as geographic location is not a defining feature for that core, and the rest of the cores were merged. (The same set of crimes is in the merged pattern regardless of whether core 16 was used for the merge.)

As these data were reconsidered by crime analysts, we found out that when these crimes were analyzed back in 2006–2007, they were viewed as two unrelated patterns, one at the end of 2006, crimes 1 to 5, and one at the beginning of 2007, crimes 6 to 10. The connection between these two subsets of crime is very subtle, and there is over a month gap between the two

Table 3. Cores and Their Defining Features for Example 1

Cores	Crimes			Geo loc	Days apart	Loc of entry	Means of entry	Premises	Ransacked	Residents	Time of day	Day	Suspect	Victim
1	1	2	3	✓	✓	✓	O	O	✓	O	✓	✓	O	O
3	1	2	5	✓	✓	✓	O	O	✓	O	✓	✓	O	O
4	1	3	4	✓	✓	✓	O	O	✓	O	✓	✓	O	O
5	1	3	5	✓	✓	✓	O	O	✓	O	✓	✓	O	O
6	1	4	5	✓	✓	✓	O	O	✓	O	✓	✓	O	O
7	1	5	6	✓	✓	✓	O	O	✓	O	✓	✓	O	O
8	2	3	4	✓	✓	✓	O	O	✓	O	✓	✓	O	O
9	2	3	5	✓	✓	✓	O	O	✓	O	✓	✓	O	✓
10	2	4	5	✓	✓	✓	✓	O	✓	O	✓	✓	O	O
11	2	4	6	✓	✓	✓	O	O	✓	O	✓	✓	O	O
12	2	5	6	✓	✓	✓	O	O	✓	O	✓	✓	O	O
13	3	4	5	✓	✓	✓	O	O	✓	O	✓	✓	O	O
14	3	5	6	✓	✓	✓	O	O	✓	O	✓	✓	O	O
15	4	5	6	✓	✓	✓	O	O	✓	O	✓	✓	O	O
16	5	6	7	O	✓	✓	O	O	✓	✓	✓	✓	O	O
17	6	8	9	✓	✓	✓	O	O	✓	O	✓	✓	O	O
18	6	8	10	✓	✓	✓	O	O	✓	O	✓	✓	O	O
19	6	9	10	✓	✓	✓	O	O	✓	O	✓	✓	O	O
20	7	8	9	✓	✓	✓	O	✓	✓	O	✓	✓	O	O
21	7	8	10	✓	✓	✓	O	✓	✓	O	✓	✓	O	O
22	7	9	10	✓	✓	✓	O	✓	✓	O	✓	✓	O	O
23	8	9	10	✓	✓	✓	O	✓	✓	O	✓	✓	O	O

patterns, so it did not occur to the crime analysts to link them. Their intuition agrees completely with the similarity graph, as the two subsets are weakly connected only by one edge; however, recall that this only describes the pattern-general similarity–what one would expect from generic pattern without considering a specific M.O. On examination of the cores, not only are they correlated in six features, but five of the cores (core indices 11, 12, 14, 15, 16) contain crimes from both of the subsets, which is strong evidence that the two subsets should be merged together. It is particularly interesting that the core consisting of crimes 5, 6, and 7 spanned the two subsets, where these crimes share the unusual feature that residents were present during the break-in.

We wondered why the offenders changed their M.O. to move a few blocks north since they committed crimes 1–6 in the same area. What may have happened is that the criminals left near the end of December (just before the holidays), and returned in February to commit housebreak number 6 in the same area as 1–5; however, they were witnessed committing crime 6 (suspect information in crime 6 reads "3 males"). This may have spooked the offenders, causing them to alter their M.O. by moving north to commit crimes 7–10. Analysts now believe that these two series were actually a single series, and that the suspect information from crime 6 can be carried through to all the crimes in the discovered series.

This is a good example to show how crime patterns can be composed of cores and exhibit similarity both in a pattern-general way and a pattern-specific way. It shows how we can use both aspects to mine patterns. This is a pattern that would be very difficult for a crime analyst to find: the M.O. changes over time, there was a long break in the middle of the series, and there was nothing deterministic (e.g., fingerprints) linking these crimes.

Table 4. Data from the Second Crime Series

No.	Date	Location of entry	Means of entry	Premises	Ransacked	Residents	Time of day	Day	Suspect	Victim
1	1/25/07	Front Door	Punched/Popped	Apartment	No	Not in	10:20–12:00	Thur	null	1 F
2	1/25/07	Unknown	Cut Screen	Apartment	No	Not in	8:45–14:30	Thur	null	1 M
3	1/25/07	Unknown	Pried	Apartment	No	Not in	9:10–21:00	Thur	null	1 F
4	1/25/07	Front door	Unlocked	Apartment	No	In	13:00–13:30	Mon	null	1 F
5	1/29/07	Front door	Key	Apartment	No	Not in	14:52–14:52	Mon	null	2 M & 3 F
6	1/29/07	Unknown	Unknown	Apartment	No	Not in	12:00–12:00	Mon	1 M	1 M
7	1/29/07	Unknown	Unknown	Apartment	No	Not in	15:00–15:00	Mon	null	2 M

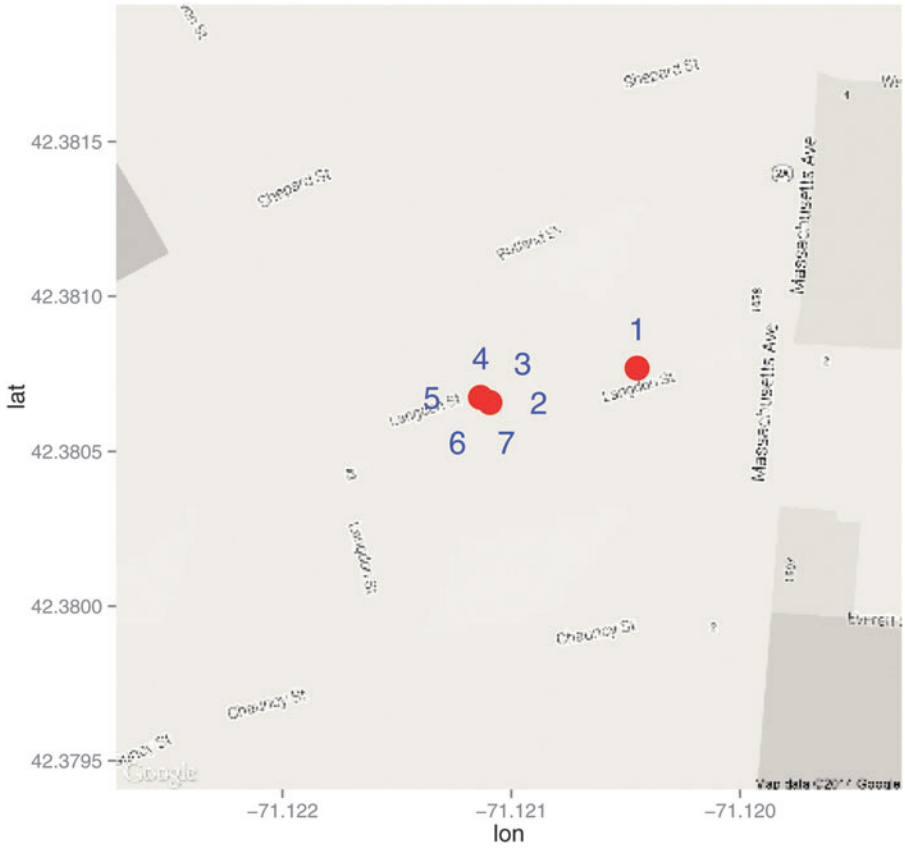

FIG. 9. The locations of crimes in the second series.

Case study 2

Table 4 and Figure 9 show a more typical pattern in 2007 discovered by our method. The crimes were committed on two dates in late January 2007, most of them in the same building. According to the Cambridge police, they arrested a suspect while he was committing the last crime in the series and confirmed that he did commit crimes 2–7. Note that the record for the fourth crime records the means of entry as "key." This is based only on the claim of a witness that the offender possesses a master key. If the offender did possess a master key to the apartments in the building, it would explain why the means of entry was unknown for other crimes in the series—the means of entry would thus have been very difficult to determine for earlier crimes where residents were not present.

Case study 3

Our method has also been used to help ensure the fidelity of the historical database of past crimes. We ran our method using data collected prior to 2006 to see whether we would be able to make discoveries.

Table 5. The Third Serieslike Pattern with $d = 5$

No.	*Date*	*Location of entry*	*Means of entry*	*Premises*	*Ransacked*	*Residents*	*Time of day*	*Day*	*Suspect*	*Victim*
1	2/9/05	Ground window	Shoved/forced	Apartment	No	In	1:47	Wed	1 White M	1 F
2	2/9/05	Rear door	Shoved/forced	Single-family House	No	In	9:50	Wed	1 Black M	1 M & 1 F
3	2/15/05	Ground window	Broke	Apartment	Yes	Not in	7:00–13:30	Tue	Null	2 M
4	2/21/05	Front door	Key	Unknown	No	Not in	7:10–10:00	Mon	Null	1 F
5	2/23/05	Front door	Pried	Apartment	No	Not in	7:10–16:00	Wed	Null	2 M
6	2/23/05	Front door	Pried	Apartment	No	Not in	7:00–14:00	Wed	Null	2 M
7	2/23/05	Front door	Pried	Apartment	No	Not in	7:45–17:25	Wed	Null	2 F
8	2/28/05	Rear door	Unknown	Apartment	No	Not in	20:55	Mon	1 White M	1 F

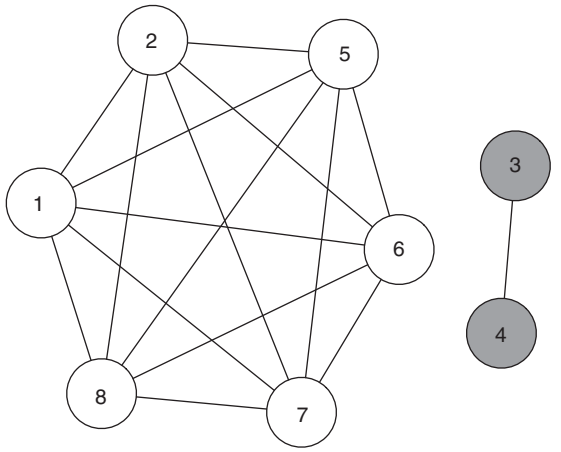

FIG. 10. Similarity graph for the third crime series.

Table 5 shows details from crimes within a 2005 pattern, discovered by both the police and our algorithm. Our algorithm and the crime analysts agreed on six out of the eight crimes in the series, but disagreed on two crimes: crime 3 and crime 4. The crime analysts identified these crimes as part of the pattern, but our algorithms did not identify these crimes as being part of the pattern. Our algorithm provides reasons why these crimes should be excluded from the pattern: They are not close to other crimes in the pattern-general sense and are not connected to the other crimes in the series within the similarity graph, as depicted in Figure 10. Neither of the crimes are contained in any cores, as shown in Table 6. In particular, the map in Figure 11 shows that these two crimes are geographically far away from the other crimes. Since geographic closeness has a large contribution to pattern-general similarity, crimes 3 and 4 are already not likely to be part of the same series. Besides that, we also notice that other aspects of crimes 3 and 4 differ from the rest of the

FIG. 11. The locations of crimes in the third series.

Table 6. Cores and Their Defining Features for Example 4

Cores	Crimes			Geographic Location	Days apart	Location of entry	Means of entry	Premises	Ransacked	Residents	Time of day	Day	Suspect	Victim
1	1	2	8	O	✓	✓	O	✓	✓	O	O	✓	O	O
2	2	5	6	O	✓	✓	O	✓	✓	O	✓	✓	O	O
3	2	5	7	O	✓	✓	O	✓	✓	O	✓	✓	O	O
4	2	5	8	O	✓	✓	O	✓	✓	O	O	✓	O	O
5	2	6	7	O	✓	✓	O	✓	✓	O	✓	✓	O	O
6	2	6	8	O	✓	✓	O	✓	✓	O	O	✓	O	O
7	2	7	8	O	✓	✓	O	✓	✓	O	O	✓	O	O
8	5	6	7	✓	✓	✓	✓	✓	✓	O	✓	✓	O	✓
9	5	6	8	✓	✓	✓	O	✓	✓	O	O	✓	O	O
10	5	7	8	✓	✓	✓	O	✓	✓	O	O	✓	O	O
11	6	7	8	✓	✓	✓	O	✓	✓	O	O	✓	O	O

pattern (see Table 5). In crime 4, the means of entry is by key, which is different from that of other crimes where entries were forced or pried. According to police narratives, the fourth crime had a 91-year-old victim, who reported that $35 was stolen from her purse by someone who used a key to enter the apartment; this indicates the crime was not part of the series, and further indicates the possibility that this crime never actually occurred. In crime 3, the apartment was ransacked while none of the other locations were. The narrative for the third crime is very generic: someone broke in through the living room window, stole jewelry and loose change, and exited through the front door. It is not clear that this crime possesses any distinguishing characteristics to identify it as being part of the series.

When examining the crime series retrospectively, the police agreed that crimes 3 and 4 likely should be excluded from the pattern (note that this pattern has not been officially verified through an arrest).

Conclusions

The problem of detecting crime series is both subtle and difficult. In cases where crime series detection is trivial, meaning that there is identifiable information about the criminal (e.g., fingerprints or DNA) then sophisticated modeling techniques are not needed—the solution is direct in those cases. The more subtle cases where crime series might otherwise become hidden in a sea of data are where data mining methods can shine. Our automated series detection methods are able to detect crime patterns within big data sets where a human could not. Detecting patterns among crime data is critical to effective policing. When a pattern is identified, police can implement effective strategies to prevent future crimes, solve past crimes, identify and capture offenders, and ensure offenders are investigated and prosecuted fully for the crimes for which they are responsible. We envision that methods such as ours could be used very broadly for crime series detection, and would be particularly useful across regions or districts, where human crime analysts would fail to be able to manually handle vast amounts of data from several different regions or data sources in order to detect patterns.

Acknowledgment

This work was supported by a grant from MIT Lincoln Laboratories.

Author Disclosure Statement

No competing financial interests exist.

References

1. Woodhams J, Bull R, Hollin CR. Chapter 6: Case linkage: identifying crimes committed by the same offender. In: Criminal Profiling: International Theory, Research, and Practice. New York: Humana Press Inc., 2007.
2. Gwinn SL, Bruce C, Cooper JP, Hick S. Exploring Crime Analysis. Readings on Essential Skills, 2nd ed. Charleston, SC: BookSurge, LLC, 2008.
3. Hering AS, Bair S. Characterizing spatial and chronological target selection of serial offenders. J R Stat Soc Ser C. 2014;63:123–140.
4. Wolfgang ME. Delinquency in a Birth Cohort. Chicago: University of Chicago Press, 1987.
5. Weisel DL. Burglary of single-family houses. Problem-Oriented Guides for Police Series, No. 18, 2002.
6. Computing Community Consortium, Computing Research Association. 2012. Challenges and opportunities with big data, a community white paper developed by leading researchers across the United States. www.cra.org/ccc/files/docs/init/bigdatawhitepaper.pdf
7. Rudin C, et al. (A working group of the American Statistical Association). 2014. Discovery with data: leveraging statistics with computer science to transform science and society. www.amstat.org/policy/pdfs/BigDataStatisticsJune2014.pdf
8. Porter MD. A statistical approach to crime linkage. ArXiv e-prints, 2014.
9. Cocx TK, Kosters WA. A distance measure for determining similarity between criminal investigations. In: Adderley R, Musgrove P, (eds.), Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining, Volume 4065 of Lecture Notes in Computer Science. Berlin: Springer, 2006, pp. 511–525.
10. Lin S, Brown DE. An outlier-based data association method for linking criminal incidents. Decis Supp Syst 2006;41:604–615.
11. Nath SV. Crime pattern detection using data mining. In: Proceedings of the Web Intelligence and Intelligent Agent Technology Workshops; 2006, pp. 41–44.
12. Singla CR, Dembla D, Chaba Y, Singla K. 2008. An optimal kd model for crime pattern detection based on semantic link analysis-a data mining

tool. www.researchgate.net/publication/228691733_An_Optimal_KD_Model_for_Crime_Pattern_Detection_Based_on_Semantic_Link_Analysis-A_Data_Mining_Tool.
13. Bennell C, Bloomfield S, Snook B, et al. Linkage analysis in cases of serial burglary: comparing the performance of university students, police professionals, and a logistic regression model. Psychol Crime Law. 2010;16:507–524.
14. Bennell C, Canter DV. Linking commercial burglaries by modus operandi: tests using regression and ROC analysis. Sci Justice. 2002;42:153–164.
15. Bennell C, Jones N. Between a ROC and a hard place: a method for linking serial burglaries by modus operandi. J Invest Psychol Offender Profiling. 2005;2:23–41.
16. Bennell C, Jones N, Melnyk T. Addressing problems with traditional crime linking methods using receiver operating characteristic analysis. Legal Criminol Psychol. 2009;14:293–310.
17. Bennell C, Snook B, Macdonald S, et al. Computerized crime linkage systems a critical review and research agenda. Criminal Justice Behav 2012;39:620–634.
18. Brown DE, Hagen S. Data association methods with applications to law enforcement. Decis Support Syst. 2003;34:369–378.
19. Ellingwood H, Mugford R, Bennell C, et al. Examining the role of similarity coefficients and the value of behavioural themes in attempts to link serial arson offences. J Invest Psychol Offender Profiling. 2012;10:1–27.
20. Markson L, Woodhams J, Bond JW. Linking serial residential burglary: comparing the utility of modus operandi behaviours, geographical proximity, and temporal proximity. J Invest Psychol Offender Profiling 2010;7:91–107.
21. Tonkin M, Santtila P, Bull R. The linking of burglary crimes using offender behaviour: testing research cross-nationally and exploring methodology. Legal Criminol Psychol. 2012;17:276–293.
22. Tonkin M, Grant T, Bond JW. To link or not to link: a test of the case linkage principles using serial car theft data. J Invest Psychol Offender Profiling. 2008;5:59–77.
23. Tonkin M, Woodhams J, Bull R, et al. Linking different types of crime using geographical and temporal proximity. Criminal Justice Behav. 2011;38:1069–1088.
24. Wang T, Rudin C, Wagner D, Sevieri R. Learning to detect patterns in crime. In: Machine Learning and Knowledge Discovery in Databases. Springer, 2013, pp. 515–530.
25. Wang T, Rudin C, Wagner D, Sevieri R. Detecting patterns of crime with series finder. In: Proceedings of AAAI Late Breaking Track; 2013.
26. Huggins JH, Rudin C. A statistical learning theory framework for supervised pattern discovery. In: Proceedings of SIAM Conference on Data Mining (SDM); 2014.
27. Adderley R. The use of data mining techniques in operational crime fighting. In: Intelligence and Security Informatics, Volume 3073 of Lecture Notes in Computer Science. Chen H, Moore R, Zeng DD, Leavitt J (eds.). Berlin: Springer, 2004, pp. 418–425.
28. Adderley A, Musgrove P. Modus operandi modelling of group offending: a data-mining case study. Int J Police Sci Manage. 2003;5:265–276.
29. Dahbur K, Muscarello T. Classification system for serial criminal patterns. Artif Intell Law. 2003;11:251–269.
30. Hering AS, Kazor K. A permutation test to identify important attributes for linking crimes of serial offenders. Stat. 2013;2:211–226.
31. Ma L, Chen Y, Huang H. Ak-modes: a weighted clustering algorithm for finding similar case subsets. In: Proceedings of the International Conference on ISKE, 2010; 2010, pp. 218–223.
32. Reich BJ, Porter MD. Partially supervised spatiotemporal clustering for burglary crime series identification. J R Stat Soc Ser A. 2014;178:465–480.
33. Santtila P, Fritzon K, Tamelander AL. Linking arson incidents on the basis of crime scene behavior. J Police Criminal Psychol. 2004;19:1–16.
34. Santtila P, Junkkila J, Kenneth Sandnabba N. Behavioural linking of stranger rapes. J Invest Psychol Offender Profiling 2005;2:87–103.
35. Sorochinski M, Gabrielle Salfati C. The consistency of inconsistency in serial homicide: patterns of behavioural change across series. J Invest Psychol Offender Profiling. 2010;7:109–136.
36. Kriegel H-P, Kröger P, Zimek A. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans Knowledge Discov Data. 2009;3:1–58.
37. Pei J, Zhang X, Cho M, et al. Maple: a fast algorithm for maximal pattern-based clustering. In: Proceedings of the International Conference on Data Mining (ICDM). New York: IEEE, 2003, pp. 259–266.
38. Wang H, Wang W, Yang J, Yu PS. Clustering by pattern similarity in large data sets. In: Proceedings of ACM SIGMOD; 2002, pp. 394–405.
39. Domeniconi C, Papadopoulos D, Gunopulos D, Ma S. Subspace clustering of high dimensional data. In: Proceedings of the SIAM Conference on Data Mining (SDM); 2004, pp. 517–521.
40. Vidal R. Subspace clustering. IEEE Signal Process Mag. 2011;28:52–68.
41. Günnemann S, Boden B, Seidl T. DB-CSC: a density-based approach for subspace clustering in graphs with feature vectors. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD). New York: Springer, 2011, pp. 565–580.
42. Moser F, Colak R, Rafiey A, Ester M. Mining cohesive patterns from graphs with feature vectors. In: Proceedings of the SIAM Conference on Data Mining (SDM); 2009, volume 9, pp. 593–604.
43. McFowland E, Speakman S, Neill DB. Fast generalized subset scan for anomalous pattern detection. J Mach Learn Res. 2013;14:1533–1561.
44. Neill DB, Cooper GF. A multivariate bayesian scan statistic for early event detection and characterization. Mach Learn. 2010;79:261–282.
45. Guan Y, Dy JG, Jordan MI. A unified probabilistic model for global and local unsupervised feature selection. In: Proceedings of the 28th International Conference on Machine Learning (ICML); 2011, pp. 1073–1080.
46. Chen H, Chung W, Xu JJ, et al. Crime data mining: a general framework and some examples. IEEE Comput. 2004;37:50–56.
47. Thongtae P, Srisuk S. An analysis of data mining applications in crime domain. In: Proceedings of the IEEE 8th International Conference on Computer and Information Technology Workshops; 2008, pp. 122–126.
48. Berk RA. Algorithmic criminology. Secur Inform. 2013;2:5.
49. Berk R. Criminal Justice Forecasts of Risk: A Machine Learning Approach. New York: Springer, 2012.
50. Berk R, Sherman L, Barnes G, et al. Forecasting murder within a population of probationers and parolees: a high stakes application of statistical learning. J R Stat Soc Ser A 2009;172:191–211.
51. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference and Prediction. New York: Springer, 2009.
52. Ghahramani Z, Heller K. Bayesian sets. In: Proceedings of Neural Information Processing Systems (NIPS); 2005.
53. Letham B, Rudin C, Heller KA. Growing a list. Data Mining Knowledge Disco. 2013;27:372–395.

Cite this article as: Wang T, Rudin C, Wagner D, Sevieri R (2015) Finding patterns with a rotten core: data mining for crime series with cores. *Big Data* 3:1, 3–21, DOI: 10.1089/big.2014.0021.

Abbreviations Used

M.O. = modus operandi
HAC = hierarchical agglomerative clustering
NN = nearest neighbor
SL = single linkage
CL = complete linkage
GA = group average
ILP = integer linear programming