Knowledge and Information Systems
https://doi.org/10.1007/s10115-018-1293-8

REGULAR PAPER

Fast network discovery on sequence data via time-aware hashing

Tara Safavi[1] · **Chandra Sripada**[2] · **Danai Koutra**[1]

Received: 21 December 2017 / Revised: 3 October 2018 / Accepted: 24 November 2018

Abstract
Discovering and analyzing networks from non-network data is a task with applications in fields as diverse as neuroscience, genomics, climate science, economics, and more. In domains where networks are discovered on multiple time series, the most common approach is to compute measures of association or similarity between all pairs of time series. The nodes in the resultant network correspond to time series, which are linked by edges weighted according to the association scores of their endpoints. Finally, the fully connected network is thresholded such that only the edges with stronger weights remain and the desired sparsity level is achieved. While this approach is feasible for small datasets, its quadratic (or higher) time complexity does not scale as the individual time series length and the number of compared series increase. Thus, to circumvent the inefficient and wasteful intermediary step of building a fully connected graph before network sparsification, we propose a fast network discovery approach based on probabilistic hashing. Our methods emphasize consecutiveness, or the intuition that time series following similar fluctuations in longer time-consecutive intervals are more similar overall. Evaluation on real data shows that our method can build graphs nearly 15 times faster than baselines (when the baselines do not run out of memory), while achieving accuracy comparable to, or better than, baselines in task-based evaluation. Furthermore, our proposals are general, modular, and may be applied to a variety of sequence similarity search tasks.

Keywords Network discovery · Brain networks · Networks · Hashing · LSH · Time series · Sequences · Knowledge discovery

1 Introduction

Prevalent among data in the natural, social, and information sciences are graphs or networks, which are data structures consisting of entities (nodes) and connections among those entities (edges). In some cases, graphs are directly observed, as in the well-studied example of online

✉ Tara Safavi
tsafavi@umich.edu

[1] Computer Science and Engineering, University of Michigan, Ann Arbor, USA

[2] Psychiatry and Philosophy, University of Michigan, Ann Arbor, USA

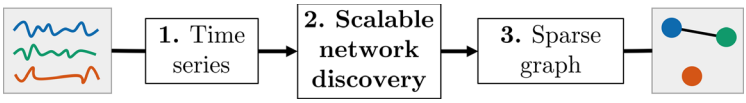


Fig. 1 Scalable network discovery. In step 2, we circumvent all-pairs similarity computations and instead only compare series that are *likely* similar

social networks, where nodes represent users and edges represent a variety of user interactions like friendship or comments. However, graphs may also be constructed from *non*-network data, a task of interest across diverse domains, which allows for powerful graph methods and tools to be readily applied toward analysis of other types of data. Network discovery on time series data in particular has many applications. For example, a common task in neuroscience is to convert a set of time series obtained via fMRI (functional magnetic resonance imaging) into a network [8,13]. Such a "network" is then used to model and analyze pairwise activity correlations among regions in the brain, ultimately for the goal of understanding brain processes like maturation and disease. Stock market time series correlation networks have also been inferred and analyzed [34]. Even "social networks" among animals may be inferred via observed co-locations or interactions over time [7]. In these fields, practitioners seek network-related insights from data that do not directly represent networked interactions. In these settings, discovered networks, which are also sometimes called association or correlation networks, connect pairs of time series (nodes) according to their pairwise similarity or association strengths.

Motivated by the growing need for scalable data analysis, we address the problem of efficient network discovery on *many* time series (Fig. 1):

Problem 1 (*Efficient network discovery on time series (informal)*) Given N univariate time series $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, efficiently construct a sparse similarity graph that captures the strong associations (edges) between pairs of time series or sequences (nodes).

Traditional network discovery on time series suffers from the simple but serious drawback of scalability. The established technique for building a graph out of N time series is to compare all pairs of series, forming a fully connected graph where nodes are time series and edges are weighted proportionally to the computed similarity or association of the nodes they connect. Afterward, the network is sparsified such that only the stronger associations, or edges with weight above a certain threshold, remain. This "all-pairs" method is at *least* an $\Omega(N^2)$ operation depending on the complexity of the time series similarity measure, which makes the process computationally inefficient on anything other than small datasets. For example, to generate even a small graph of five thousand nodes, about 12.5 million comparisons are required, where each comparison itself is at least linear in the time series length: the popular Euclidean distance and correlation measures are linear, and the dynamic time warping (DTW) distance measures are slower yet, adding an extra runtime factor as the series length increases. Furthermore, the network may eventually lose most of its edges via thresholding before further analysis, rendering many of the original comparisons wasteful.

We propose to circumvent the bottleneck of the established network discovery approach, all-pairs sequence comparison, by introducing a new locality-sensitive hashing method that quickly identifies series with similar time-consecutive fluctuations. In our approach, we first compute a compact randomized signature for each time series, then hash all series with the same signature to the same "bucket" such that only the intra-bucket pairwise similarity scores need be computed.

Contributions Our main contributions are as follows:

- *Novel sequence similarity measure and corresponding metric* Motivated by the popularity of correlation as an association measure, we propose ABC, a novel, intuitive, and generalizable time series similarity measure. Our similarity measure, ABC, captures time-consecutive matching fluctuations between time series. To use ABC in conjunction with locality-sensitive hashing (LSH), which requires a distance *metric* to provide theoretical guarantees on the similarity search process (Sect. 5), we show that ABC has a corresponding distance metric. To the best of our knowledge, ABC is the first similarity measure that both quantifies consecutiveness in time series trends and has a corresponding metric.
- *Network discovery via locality-sensitive hashing* Using the theoretical foundations of ABC, we introduce a new family of LSH hash functions, ABC-LSH. We show how the false positive and negative rates of the randomized hashing process can be controlled in network discovery.
- *Evaluation on real data* We evaluate the efficiency, accuracy, and robustness of our proposals. To evaluate accuracy, we rely on domain knowledge from neuroscience, an area of active research on discovered networks. The graphs built by our ABC variants are created up to 15 times faster than baselines, while performing as well or better in classification-based evaluation.

Outline The remainder of this work is organized as follows: In Sect. 2, we review related work in network discovery and similarity search. Section 3 gives a high-level overview of the problem and our proposed solution. In Sect. 4, we detail our proposed similarity measure, ABC, and in Sect. 5 we use ABC to design a new locality-sensitive hashing family, ABC-LSH. We review our proposals in Sect. 6 and enumerate and analyze our experimental results in Sect. 7. Finally, we conclude with discussions of our work in Sects. 8 and 9.

2 Related work

We briefly review the related literature in network structure discovery, nearest-neighbor search, and locality-sensitive hashing. In summary, while problems tangential to our addressed task have been explored, some to a greater degree than others, to the best of our knowledge efficient network discovery on time series with locality-sensitive hashing has not been explored.

2.1 Network discovery

The field of *network discovery* or *network inference* concerns constructing network representations from indirect, possibly noisy measurements with unobserved interactions [7]. For example, functional connectivity, which models the brain as a network constructed from functional magnetic resonance imaging (fMRI), is an area of intense recent interest in neuroscience [8]. The goal of functional connectivity is to identify network-theoretical properties, like the network clustering coefficient or average path length, that indicate brain health, disease, or development. Similarly, network discovery is of interest in other domains that collect data via monitoring or sensors, like genomics, climate science, finance, transportation, and ecology. Such discovered networks serve a variety of knowledge discovery tasks, like anomaly detection [1], summarization [30,39], prediction [32], inference and similarity [25,26,41].

Typically, practitioners in these domains infer interaction networks using direct measures of association like correlation. However, recently there has been increased interest in inferring (potentially time-varying) graphical models from multivariate data using lasso regularization [16,42]. These approaches assume that the data follow a k-variate normal distribution $N(0, \Sigma)$, where k is the number of parameters and Σ is the covariance of the distribution. While recent work focuses on scaling this approach, we tackle efficient network discovery *without* distributional assumptions on the edges of the discovered network.

The related field of *graph signal processing* (GSP) addresses graph representations of high-dimensional signal data [40]. As GSP involves constructing association networks from signal data, its output is the same as network discovery. However, GSP's focus is not on the efficiency or structural evaluation of discovered networks. Rather, its goal is to extend traditional signal processing tools, like signal filtering and transformations, to graphs.

2.2 Nearest-neighbor search

The problem of finding nearest neighbors has been addressed from several perspectives. In a *k-nearest-neighbor* (k-NN) graph, each node is connected via a directed edge to the top k most similar other nodes in the graph. Some techniques proposed to improve the quadratic runtime of traditional k-NN graph construction include local search algorithms and hashing [6,14,44]. However, limiting the number of neighbors per node is an unintuitive task in our case. For this reason, our proposed method lets the hashing process determine which pairs of nodes are connected. Moreover, the output data structure of k-NN graph discovery algorithms fundamentally differs from ours, as we seek to discover an *undirected*, *weighted* graph.

Related is the problem of the ϵ-nearest-neighbor (ϵ-NN) graph, in which all node pairs above a similarity score ϵ are connected via undirected edges. The ϵ-NN graph is a variant of the well-studied *set similarity self-join problem* [6,10], which seeks to identify all pairs of objects above a user-set similarity threshold. While there has been significant work in speeding up this approach, we again make a case against thresholding as a central step in network discovery. Indeed, practitioners in domains like neuroscience have noted that the (potentially ad-hoc) choice of threshold can significantly affect the resultant graph structure [5]. Therefore, we seek to analyze the output network's connectivity patterns and strengths without hard-to-define edge-weight thresholds. Although there has been recent advancement in scalable time series subsequence self-join *without* thresholding [43], such efforts find the nearest neighbor of every subsequence in a given time series. Our network discovery task is neither concerned with time series subsequences nor constrained to single nearest-neighbor search.

More recently, Scharwächter et al. [38] propose COREQ, a fast method for approximating the full correlation matrix using triangular bounds. While the goal of COREQ is similar to ours—avoiding computation of all N^2 correlations between time series—the key difference is that we focus on efficiently computing only the *strongest* associations between time series to yield a sparse network. By contrast, COREQ *approximates* all correlations, weak or strong, between pairs of time series below a specified error threshold.

2.3 Locality-sensitive hashing

Locality-sensitive hashing (LSH) has been successfully employed in various settings, including efficient discovery of similar documents and alignment of multiple networks [17]. Unlike general hashing, which aims to avoid collisions between data points, LSH *encourages* collisions between items such that colliding elements are similar with high probability [2]. LSH

provides formal guarantees on the probability of false negatives and positives in approximate similarity search given a distance *metric* (i.e., a distance measure that satisfies the triangle inequality) and an associated family of LSH functions [28]. We discuss more technical details of LSH in Sect. 5.

A few general methods of hashing time series have been proposed, although neither for the purpose of network discovery nor for capturing time-ordered similarity between time series [21,23,24]. Most recently, random projections of sliding windows on time series have been proposed for constructing approximate short hash signatures [31]. However, this approach uses dynamic time warping (DTW) as a measure of time series distance. While DTW and other nonlinear alignment schemes have the advantage of matching similarly shaped time series out of phase in the time axis, and may empirically work well with hashing, such measures cannot be *metrics* by definition [33]. Therefore, we do not consider these measures. Without a metric, we lack the theoretical foundation for true LSH and cannot provide guarantees on false positive and negative rates.

3 Overview of problem and approach

The problem we address is given as:

Problem 2 (*Multiple time series to weighted graph*) Given N time series $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, construct a sparse similarity graph where each node corresponds to a time series $\mathbf{x}^{(i)}$ and each edge is weighted according to the association of the nodes $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ it connects.

As previously stated, traditional network discovery on time series is quadratic in the number of time series N. Its total complexity also depends on the chosen similarity measure. We thus propose a modular three-step approach that circumvents the costly all-pairs comparison step (Fig. 2):

1. *Preprocess time series* First, the input real-valued time series are approximated as binary sequences to capture just their fluctuations.
2. *Hash binary sequences to buckets* Next, the binary sequences are hashed to short, randomized signatures in a "time-aware" fashion. To achieve this, we define a novel similarity measure, ABC, that quantifies time-consecutive similarity in sequences. Beyond being qualitatively comparable to correlation, ABC is theoretically eligible for LSH, as it has a corresponding distance metric. It also addresses some shortcomings of pointwise comparison measures, which we illustrate in Sect. 4. We show that ABC's complementary distance measure is a metric, and use this result to design an LSH family tailored to capturing time-consecutive similarity.

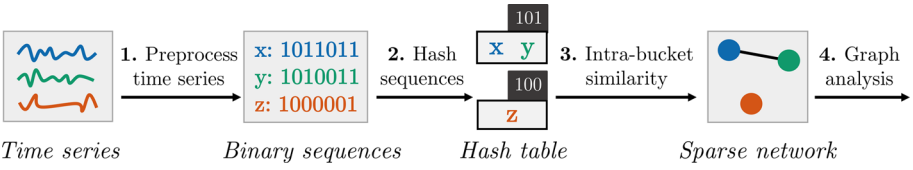


Fig. 2 Proposed network discovery method, ABC-LSH. The output of step 3 is a graph in which edges are weighted according to node (time series) similarity

3. *Compute intra-bucket pairwise similarity* The similarity between each pair of time series that hash to the same signature, or bucket, is computed. A weighted edge is created between each pair of colliding time series.

The output of this process is a graph in which all pairs of time series that collide in any round of hashing are connected by an edge weighted according to their similarity. For reference, we define our major symbols in Table 1.

4 ABC: quantifying time-consecutive similarity

The motivation behind our proposed measure, ABC, is that capturing similarity via *pointwise* agreement—for example, Euclidean distance or other popular measures—can be ineffective, especially when combined with approximate similarity search techniques like hashing. As shown in Fig. 3, agreement between two series in t randomly scattered timesteps does not always capture true similarity in trends. By contrast, two series following the same pattern of fluctuations in t *consecutive* timesteps are arguably more associated. As Iglesias and Kastner [18] note, although Euclidean distance is often sufficient in time series data mining applications, it is in principle invariant with respect to changes in time ordering among pairs of time series and thus "blind" to capturing time-ordered similarity.

Example 1 Consider the three time series in Fig. 3. Here, the (z-normalized) time series $\mathbf{x}$ and $\mathbf{y}$ are clearly more similar to each other than to $\mathbf{z}$, which does not fluctuate at all within the time window. However, the Euclidean distance scores are highest between $\mathbf{x}$ and $\mathbf{y}$—in other words, $\mathbf{x}$ and $\mathbf{y}$ are deemed the *least* similar—due to the nature of pointwise distance comparison. By contrast, our ABC metric (Sect. 4.2) correctly assigns the lowest distance score between $\mathbf{x}$ and $\mathbf{y}$, as it quantifies the matching fluctuation trends between the series. In doing so, it also "corrects" the small phase misalignment between $\mathbf{x}$ and $\mathbf{y}$, although we note that time series phase alignment is *not* the goal of ABC (see Sects. 2.3 and 8 for more discussion).

Table 1 Major symbols

Symbol	Definition
$\mathbf{x}$	A time series, or a sequence of n real values
$\mathbf{X}$	A set of N time series $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, each of length n
$\mathbf{b}(\mathbf{x})$	The binary approximation of a time series $\mathbf{x}$
$S(\mathbf{x}, \mathbf{y})$	The maximum ABC similarity between two sequences
p	The number of agreeing runs between two sequences
k_i	The length of the i-th agreeing run between two sequences
α	The parameter upon which ABC operates; α controls the emphasis on consecutiveness as a factor in similarity scoring
$\mathbf{F}$	A locality-sensitive family of hash functions
k	The length of a window or subsequence of $\mathbf{x}$
r	The number of hash functions to AND with LSH
b	The number of hash functions to OR with LSH

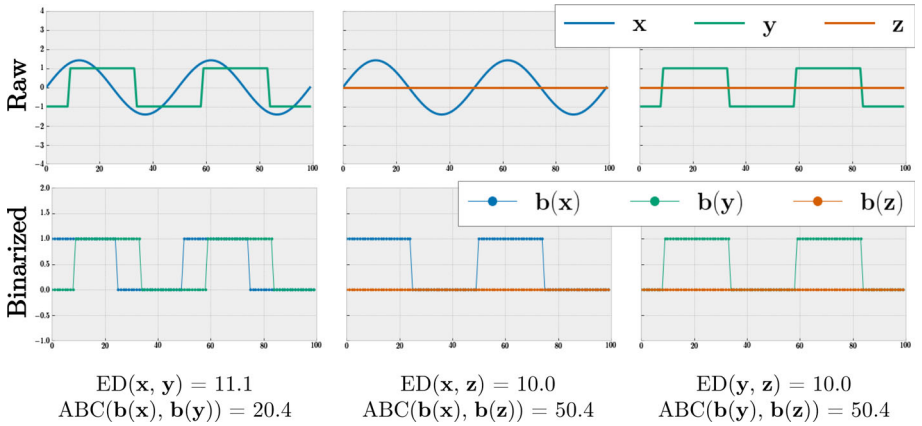


Fig. 3 Capturing consecutiveness can yield better results. Although the time series $\mathbf{x}$ and $\mathbf{y}$ (top left) are the most similar, the Euclidean distance pointwise comparison method fails to reflect this. Here, $\mathbf{x}$ and $\mathbf{y}$ are assigned the highest Euclidean distance and thus the lowest similarity score. By contrast, our ABC metric, computed here with $\alpha = 10^{-4}$, correctly assigns a much lower distance score to $\mathbf{x}$ and $\mathbf{y}$

In the following sections, we outline preliminaries for ABC, define ABC similarity and distance, and analyze some of ABC's important properties. Finally, we generalize ABC to other data types.

4.1 Preprocessing: time series representation

The first step in our pipeline converts raw, real-valued time series to binary, easily hashable sequences. If the series are already discretized, this step can be skipped (Sect. 4.4).

In the data mining literature, several binarized representations of time series have been proposed [3,22,36]. The one we use has been called the "clipped" representation of a time series [36].

Definition 1 (*Binarized representation of time series*) Given a time series $\mathbf{x} \in \mathbb{R}^n$, the binarized representation of the series $\mathbf{b}(\mathbf{x})$ replaces each real value x_i, $i \in [1, n]$, by a single bit such that $\mathbf{b}(x_i) = 1$ if x_i is above the series mean $\mu_{\mathbf{x}}$, and 0 otherwise.

We choose this representation because it captures key fluctuations in the time series—which we want to compare, as correlation does—while providing an approximation suitable for fast similarity search. In particular, LSH requires a method of constructing hash signatures of reduced dimensionality from input data points: this method depends on the similarity or distance measure used to compare the data points. As we show in the following sections, binarizing the time series naturally facilitates the construction of short, representative hash signatures, while still retaining information naturally encoded in time. We demonstrate in Sect. 7.4 that these benefits outweigh the loss of information in binarization. Binarizing the time series enables fast network discovery via hashing while maintaining accuracy. We are able to achieve comparable, or in some cases even *better*, accuracy than correlation with binarized time series.

4.2 ABC: approximate binary correlation

Given binary sequences that approximate real-valued time series, we propose an intuitive and simple measure of similarity and a complementary *metric* (Sects. 5.1, 1) to quantify matching consecutive fluctuations between pairs of series. To the best of our knowledge, ABC is the first similarity measure with an associated metric that both explicitly quantifies consecutive similarity and admits LSH (Sect. 5.2).

The intuition behind our proposed similarity measure, ABC or Approximate Binary Correlation, is to count matching bits between two binarized time series $\mathbf{x}$ and $\mathbf{y}$. In doing so, we slightly exponentially *weight* consecutively agreeing bits in $\mathbf{x}$ and $\mathbf{y}$ such that the longer the consecutive matching subsequences, which Balakrishnan and Koutras [4] call *runs*, the more similar $\mathbf{x}$ and $\mathbf{y}$ are deemed.

More formally, we define ABC similarity as a summation of multiple geometric series, which elegantly captures this intuition. For some $0 < \alpha \ll 1$, ABC adds $(1+\alpha)^i$ to the total "similarity score" for every i-th consecutive element of agreement between the two sequences, starting with $i = 0$ each time a new run begins. Thus, $\mathbf{x}$ and $\mathbf{y}$'s similarity is a sum of $1 \leq p \leq \frac{n}{2}$ geometric series, each with a common ratio $r = (1+\alpha)$ and a length k_i where $k_1 + \cdots + k_i + \cdots + k_p \leq n$. In practice, these matching subsequences can be identified via a linear scan of $\mathbf{x}$ and $\mathbf{y}$ by keeping a counter variable that is reset before each new run begins and incremented as matching bits are identified.

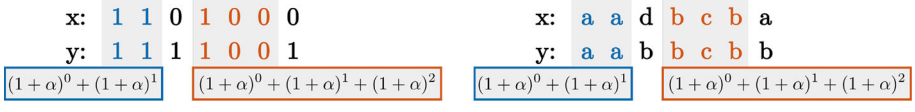
Definition 2 (*ABC (Approximate Binary Correlation) similarity*) Given two binary sequences $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ that have p matching consecutive subsequences i of length k_i, the ABC similarity is defined as

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} \sum_{b=0}^{k_i - 1} (1+a)^b = \frac{\sum_{i=1}^{p} (1+\alpha)^{k_i} - p}{\alpha} \tag{1}$$

where $\alpha \in (0, 1]$ controls the emphasis on consecutiveness: the higher the α, the higher this emphasis. Above we use that the sum of a geometric progression is $\sum_{b=0}^{n-1} r^b = \frac{1-r^n}{1-r}$ for $r \neq 1$.

Example 2 Consider $\mathbf{x} = \mathbf{1101000}$ and $\mathbf{y} = \mathbf{1111001}$ (Fig. 4a). We add $(1+\alpha)^0$ for the first bit of agreement and $(1+\alpha)^1$ for the second bit of agreement. The series do not agree in the third bit, so we resume increasing the total with the next agreeing bit, adding $(1+\alpha)^0$ for the fourth bit, $(1+\alpha)^1$ for the fifth bit, and $(1+\alpha)^2$ for the sixth bit. The total similarity is $s(\mathbf{x}, \mathbf{y}) = 1 + (1+\alpha) + 1 + (1+\alpha) + (1+\alpha)^2$. With $\alpha = 10^{-3}$, the ABC similarity is $s(\mathbf{x}, \mathbf{y}) = 1 + 1.001 + 1 + 1.001 + 1.001^2 \approx 5.004$, which is slightly higher than the number of agreeing bits in the series, as expected.

ABC distance We denote the maximum possible ABC similarity, which occurs when two sequences are identical, as $S(\mathbf{x}, \mathbf{y})$. This value is the sum of a geometric progression from 0



(a) ABC on binarized data. **(b)** ABC on general sequence data.

Fig. 4 ABC similarity. In both examples, the ABC similarity between $\mathbf{x}$ and $\mathbf{y}$ is the sum of two geometric series that encode length-2 and 3 runs, respectively

to $n-1$ with common ratio $(1+\alpha)$: $S(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n-1}(1+\alpha)^i = \frac{(1+\alpha)^n - 1}{\alpha}$. To normalize ABC similarity within the range of 0 and 1, we can divide the observed ABC similarity between two sequences $\mathbf{x}$ and $\mathbf{y}$ by S. In the example in Fig. 4, the similarity ratio is $\frac{5.004}{7.002} \approx 0.712$.

The complementary ABC *distance* score $d(\mathbf{x}, \mathbf{y})$ is easily derived by subtracting the observed similarity score s from the maximum similarity S. By this definition, identical sequences have a distance of 0, and sequences with no agreeing bits have a distance of $S(\mathbf{x}, \mathbf{y})$.

Definition 3 (*ABC distance*) Given two binary sequences $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ that have p matching consecutive subsequences i of length k_i, the ABC distance is defined as

$$d(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) - s(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{p}(1+\alpha)^{k_i} + p - 1}{\alpha} \tag{2}$$

4.3 Understanding ABC similarity

In this section, we further analyze our proposed similarity measure, ABC. We discuss how to control its single parameter α and compare ABC to correlation, which is widely used as an association measure in network discovery.

Choosing α We recommend choosing α with regard to two criteria. The first criterion is the desired emphasis on consecutiveness in similarity scoring. For example, choosing $\alpha = 0$ reduces the similarity score to the complement of Hamming distance. Increasing α both increases the maximum similarity score S (Sect. 4.2) and the "gap" between pairs of series that match in long consecutive runs versus shorter runs—for example, agreement in every other timestep.

The second criterion is the length of the compared sequences. The longer the sequences, the smaller α should be to avoid very large numbers in exponentiation. For example, for time series of length $10,000$, choosing $\alpha = 10^{-5}$ results in a maximum exponent of $1.00001^{10000} \approx 1.105$ for two sequences in perfect agreement. By contrast, choosing $\alpha = 10^{-3}$ results in a maximum exponent of $1.001^{10000} \approx 21916.68$ for the same two sequences.

Comparison to correlation One of our objectives in designing ABC is to assign similarity scores comparable to Pearson's correlation coefficient, since the latter is the most common association measure in network discovery on time series [8,27]. Intuitively, ABC is like correlation in that it assigns higher similarity to pairs of series following similar trends, which occurs when the series follow the same pattern over longer consecutive intervals.

To confirm that ABC is indeed similar to correlation, we compared Pearson's correlation coefficient to normalized ABC similarity with $\alpha = 10^{-4}$ on all pairs of time series in 10 brain scans from the COBRE dataset (Sect. 7.1). We performed linear regression on all score pairs, finding a strong linear relationship—on average, $r = 0.84$ with a p-value of 0—between the correlation and normalized ABC scores (Fig. 5).

However, while we observe a strong relationship between ABC similarity and correlation, the ABC similarity measure as given in Definition 2 does not take into account inversely correlated relationships as Pearson's correlation coefficient does. In fact, our original definition of ABC assigns low similarities to pairs of sequences that display anti-correlations, and a similarity score of 0 to two complementary binary sequences—in other words, sequences that are perfectly inversely correlated.

Some domains interested in network discovery take absolute-valued Pearson's correlation coefficient between time series as the network edge weights, thereby keeping both the strong

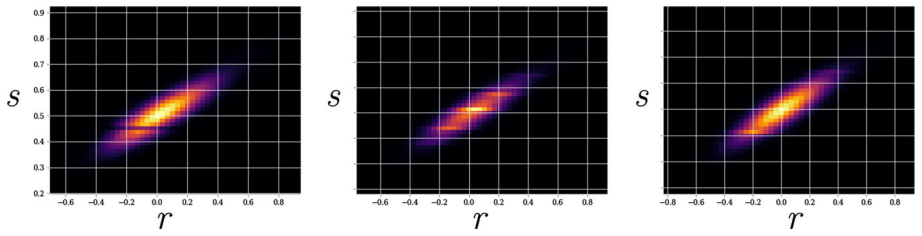


Fig. 5 Correlation versus ABC. Pearson's correlation (x-axis) versus normalized ABC similarity (y-axis) scores for all pairs of time series from three brain scans

positive and negative correlations [27]. To address this, we introduce a simple measure of "anti-correlation" based on our previous definition of ABC similarity:

Definition 4 (*Inverse ABC similarity*) Given two binary sequences $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$, the inverse ABC similarity score is computed as given in Definition 2, but on *disagreeing* runs—subsequences in which all bits differ—rather than *agreeing* runs.

Take the previous example in Fig. 4a. Sequences $\mathbf{x}$ and $\mathbf{y}$ only disagree in the third and seventh bits, so their inverse ABC score is $2(1+\alpha)^0 = 2$.

In practice, we need not compute separate similarity scores for ABC and inverse ABC per pair of sequences. It suffices to simply keep two running totals while scanning the sequences. For each agreeing pair of bits, the ABC running total score increases as specified by Definition 2. For each disagreeing pair of bits, the inverse ABC total score increases as specified by Definition 4.

4.4 Generalizing ABC

While we propose and evaluate ABC as a similarity measure operating on binary sequences, ABC may in principle be applied to sequences of any data type given an indicator function between elements x_i in sequence $\mathbf{x}$ and y_i in sequence $\mathbf{y}$:

$$\delta_{x_i y_i} = \begin{cases} 1 & \text{if} \quad x_i = y_i \\ 0 & \text{otherwise} \end{cases}$$

This function outputs 1 if the corresponding sequence elements agree and 0 otherwise.

Example 3 Assuming time series input, one may wish to discretize the series using the well-known SAX symbolic representation [29]. Given $\mathbf{x} = \mathbf{aadbcba}$ and $\mathbf{y} = \mathbf{aabbcbb}$ (Fig. 4b), we can use $\delta_{x_i y_i}$ to exponentially weight consecutively matching *symbols* in the sequences. As in our earlier example with binary sequences, we add $(1+\alpha)^0 + (1+\alpha)^1$ for the first two agreeing symbols, then $(1+\alpha)^0 + (1+\alpha)^1 + (1+\alpha)^2$ for the fourth, fifth, and sixth symbols, respectively. The total ABC similarity between $\mathbf{x}$ and $\mathbf{y}$ is $s(\mathbf{x}, \mathbf{y}) = 2 + 2(1+\alpha) + (1+\alpha)^2$. Our window LSH family (Sect. 5.2) is easily extendable to such data.

In theory, even similarity between pairs of real-valued time series can be computed with ABC. For example, with an ϵ such that values within $|\epsilon|$ are considered "equal", ABC can be computed with an indicator function such as:

$$\delta_{x_i y_i} = \begin{cases} 1 & \text{if} \quad |x_i - y_i| \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

However, unlike our mean-based proposal, this approach requires normalizing the time series beforehand to ensure matching scales. Furthermore, our window LSH family requires a finite vocabulary of symbols and thus does not allow "soft equality" between sequence elements. We leave extensions of this nature for future work.

5 Scaling ABC using LSH

Beyond proposing ABC as a standalone sequence similarity measure, we apply ABC similarity and its distance complement to LSH for the ultimate goal of fast network discovery. In doing so, we prove that ABC distance is a metric, and use this result to design a new locality-sensitive hashing family tailored to capturing consecutive ("time-aware") similarity with ABC.

5.1 Theoretical foundation: metrics

The first step in designing an LSH family is to show that the distance measure in question is a *metric*, since LSH families may only be constructed for distance metrics (although note that not every distance metric has a corresponding LSH family). Upholding the metric axioms allows for guarantees of false positive and negative rates in hashing. A metric is defined as follows:

Definition 5 (*Metric*) A metric is a distance measure that satisfies the following axioms:

1. *Identity* $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$.
2. *Non-negativity* $d(\mathbf{x}, \mathbf{y}) \geq 0$.
3. *Symmetry* $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$.
4. *Triangle inequality* $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$.

Our main result is the following:

Theorem 1 (ABC distance is a metric) *The ABC distance measure (Definition 3) is a metric. It satisfies all the metric axioms, including the triangle inequality. By extension, the distance based on ABC's inverse similarity measure (Definition 4) is also a metric.*

Proof We give a sketch of the proof that ABC distance satisfies these properties, with necessary supporting proofs in the appendix.

1. *Identity* Following Definition 3, the ABC distance $d(\mathbf{x}, \mathbf{y})$ is 0 when $p = 1$ and $k_1 = n$. This occurs when $\mathbf{x}$ and $\mathbf{y}$ share a single run of length n, which means that $\mathbf{x} = \mathbf{y}$. Likewise, when $\mathbf{x} = \mathbf{y}$, the number of agreeing runs p is 1 and the run length is n, so $d(\mathbf{x}, \mathbf{y}) = 0$.
2. *Non-negativity* If $\mathbf{x}$ and $\mathbf{y}$ are the same, $d(\mathbf{x}, \mathbf{y}) = 0$. Otherwise, it must be that $S(\mathbf{x}, \mathbf{y}) > s(\mathbf{x}, \mathbf{y})$ (see "Appendix B.1"), so $d(\mathbf{x}, \mathbf{y}) \geq 0$.
3. *Symmetry* The sequence comparison order does not change the distance.
4. *Triangle inequality* This is the most complex and difficult-to-satisfy property. We prove that ABC distance satisfies the triangle inequality by induction, considering different options for agreement between sequences and showing that the triangle inequality always holds. We refer the reader to the full proof in "Appendix B.2". □

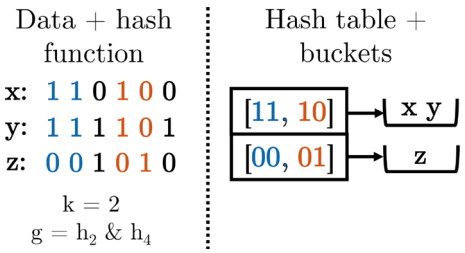


Fig. 6 Window hashing with $\mathbf{F}_W$. The hash signatures are the concatenation (i.e., the AND) of the length-2 windows starting from the first and fourth bits of each sequence

5.2 ABC-LSH definition

Although not all metrics have a corresponding LSH family, the ABC distance does. We begin by defining an LSH family:

Definition 6 (*Locality-sensitive family of hash functions* [28]) Given some distance measure $d(\mathbf{x}, \mathbf{y})$ satisfying the metric axioms, a family of locality-sensitive hash functions $\mathbf{F} = (h_1(\mathbf{x}), \ldots, h_f(\mathbf{x}))$ is said to be (d_1, d_2, p_1, p_2)-sensitive if for every function $h_i(\mathbf{x})$ in $\mathbf{F}$ and two distances $d_1 < d_2$:

1. If $d(\mathbf{x}, \mathbf{y}) \leq d_1$, the probability that $h_i(\mathbf{x}) = h_i(\mathbf{y})$ is at least p_1. The higher the p_1, the lower the probability of false negatives.
2. If $d(\mathbf{x}, \mathbf{y}) \geq d_2$, the probability that $h_i(\mathbf{x}) = h_i(\mathbf{y})$ is at most p_2. The lower the p_2, the lower the probability of false positives.

The simplest LSH family uses bit sampling [19] and applies to Hamming distance, which quantifies the number of differing components between two vectors. The bit-sampling LSH family $\mathbf{F}_H$ over n-dimensional binary vectors consists of all functions that randomly select one of its n components or bits: $\mathbf{F}_H = \{h : \{0,1\}^n \rightarrow \{0,1\} \mid h(\mathbf{x}) = x_i \text{ for } i \in [1, n]\}$. Under this family, $h_i(\mathbf{x}) = h_i(\mathbf{y})$ if and only if $x_i = y_i$. In other words, the i-th bit of $\mathbf{x}$ must be the same as the i-th bit of $\mathbf{y}$. The $\mathbf{F}_H$ family is a $(d_1, d_2, 1 - \frac{d_1}{n}, 1 - \frac{d_2}{n})$-sensitive family. Here, p_1 describes the probability of two vectors colliding when their distance is at most d_1 (i.e., $\mathbf{x}$ and $\mathbf{y}$ differ in at most d_1 bits). Thus, p_1 corresponds to the *complement* of the probability of selecting one of the disagreeing bits out of the total n bits. The probability p_2 is similarly derived.

We propose a new LSH family $\mathbf{F}_W$, extending our emphasis on consecutiveness to hashing. While the established LSH family on Hamming distance samples bits, our proposed LSH family $\mathbf{F}_W$ consists of randomly sampled windows (subsequences), starting from the same index, for all sequences in the dataset (Fig. 6).

Theorem 2 (Window sampling LSH family) *Given a window size k, our proposed family of hash functions* $\boldsymbol{F}_W$ *consists of* $n - k + 1$ *hash functions:*

$$\boldsymbol{F}_W = \{h : \{0,1\}^n \rightarrow \{0,1\}^k \mid h(\mathbf{x}) = (x_i, \ldots, x_{i+k-1}), i \in [1, n-k+1]\}$$

Equivalently, $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ *if and only if* $(x_i, \ldots, x_{i+k-1}) = (y_i, \ldots, y_{i+k-1})$. *Using ABC distance, the locality-sensitive family* $\boldsymbol{F}_W$ *is* $(d_1, d_2, 1-\alpha\frac{d_1}{(1+\alpha)^n-1}, 1-\alpha\frac{d_2}{(1+\alpha)^n-1})$*-sensitive.*

Proof The probabilities p_1 and p_2 are derived by normalizing d_1 and d_2 and taking their complement, the same way that p_1 and p_2 are derived for the Hamming distance LSH family. In the case of ABC distance, we normalize both d_1 and d_2 in the range $[0, 1]$ by dividing by $S(\mathbf{x}, \mathbf{y})$. We then take their complement to obtain $p_1 = 1 - \alpha\frac{d_1}{(1+\alpha)^n-1}$ and $p_2 = 1 - \alpha\frac{d_2}{(1+\alpha)^n-1}$. □

5.3 Controlling false positives and negatives

Given an LSH family, it is typical to construct new "amplified" families by the AND and OR constructions of $\mathbf{F}$ [28], which provide control of the false positive and negative rates in the hashing process. Concretely:

Definition 7 (*LSH AND and OR constructions*) Given a locality-sensitive family of hash functions $\mathbf{F} = (h_1(\mathbf{x}), \ldots, h_f(\mathbf{x}))$, the **AND construction** creates a new hash function $g(\mathbf{x})$ as a logical AND of r members of $\mathbf{F}$. The new hash function $g(\mathbf{x})$ consists of $\{h_i\}^r$, each i chosen uniformly at random without replacement from $[1, f]$. Then $g(\mathbf{x}) = g(\mathbf{y})$ if and only if $h_i(\mathbf{x}) = h_i(\mathbf{y})$ *for all* $i \in [1, r]$. The **OR construction** constitutes a logical OR of b hash functions in $\mathbf{F}$. In this case, we say that $g(\mathbf{x}) = g(\mathbf{y})$ if $h_i(\mathbf{x}) = h_i(\mathbf{y})$ *for any* $i \in [1, b]$.

Below, we detail the effects of the AND and OR constructions on $\mathbf{F}_W$.

AND operation We have a single hash table and a hash function $g(\mathbf{x}) = \{h_i\}^r$. Each h_i is chosen uniformly at random from $\mathbf{F}_W$ (Fig. 6). We compute a hash signature for each data point $\mathbf{x}^{(i)}$ as a concatenation of the r (potentially overlapping) length-k windows starting at index i for all $h_i \in g$. The AND operation turns any locality-sensitive family of hash functions $\mathbf{F}$ into a new $(d_1, d_2, {p_1}^r, {p_2}^r)$-sensitive family $\mathbf{F}$'.

OR operation We have a hash function $g(\mathbf{x}) = \{h_i\}^b$. Again, each h_i, chosen uniformly at random from $\mathbf{F}_W$, specifies a length-k window starting at index i. We create b hash tables. In the j-th round of hashing, $j \in [1, b]$, we compute hash signatures for all data points using the j-th hash function $h_i \in g$. Thus each data point is hashed b times. The OR operation turns the same family $\mathbf{F}$ into a $(d_1, d_2, 1 - (1 - p_1)^b, 1 - (1 - p_2)^b)$-sensitive family.

Parameters As we show in the experiments, parameter setting is straightforward based on the desired level of approximation or preciseness in the network discovery process. The higher the r, the lower the probability of false positives, as more windows are sampled and hashing becomes more precise. Conversely, the higher the b, the lower the probability of false negatives, as more sequences are given the opportunity to collide. In all cases—AND, OR, both, or neither—an edge between each colliding pair $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is added to the discovered graph G.

Regardless of parameter values, we cannot make strict runtime guarantees with ABC-LSH, since the number of hash collisions depends on the nature of the data. However, while in theory hashing-based network discovery is worst-case $O(n^2 t)$ (i.e., every length-t time series is exactly the same), we show in the experiments that most datasets—even those with high average pairwise correlation—result in sub-quadratic network discovery time.

6 Putting everything together

Having introduced a new metric, ABC distance, and its corresponding LSH family ABC-LSH, we turn back to our original goal of scalable network discovery.

ABC-LSH pipeline We review the ABC-LSH pipeline in Algorithm 1. Given N time series, each of length n, we convert all data to binary following the representation in Sect. 4.1 (line 1 of Algorithm 1). We then apply the subsequence hashing scheme of $\mathbf{F}_W$ (Sect. 5.2), setting r and b (Sect. 5.3) according to the desired false positive and negative rates (line 2). Finally, we compute all intra-bucket pairwise sequence similarities to construct a sparse weighted network for further analysis (lines 3–9).

Algorithm 1: Network discovery with ABC-LSH

Input : A set of N time series $\mathbf{X}$;
k: length of the window to sample;
r: the number of windows per sequence signature;
b: the number of hash tables to construct

Output: A weighted graph $G = (V, E)$ where each node $\mathbf{x}^{(i)} \in V$ is a time series and each edge $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \in E$ has weight ABC($\mathbf{b}(\mathbf{x}^{(i)})$, $\mathbf{b}(\mathbf{x}^{(j)})$)

```
// Step 1: preprocess time series
X ← BINARIZE(X)
// Step 2: fast sequence similarity search
buckets ← LSH- AND(X, k, r) ∪ LSH- OR(X, k, b)
// Step 3: build sparse weighted network
G ← GRAPH
for bucket ∈ buckets do
    for (x^(i), x^(j)) ∈ bucket do
        weight ← ABC(x^(i), x^(j))
        G.ADDEDGE(x^(i), x^(j), weight)
    end
end
return G
```

Further optimization There are several opportunities for optimization within the ABC-LSH pipeline. For one, the hashing stage is trivially parallelizable on several levels. Each of the b hash tables is independent and thus may be independently constructed and processed. Furthermore, each bucket within each hash table is independent, so all bucket-level similarity computations can occur in parallel.

Memoization, which trades computation for memory usage, may also be employed. If there are multiple rounds of hashing, the same pair $(\mathbf{x}, \mathbf{y})$ may collide more than once. In this case, a lookup table storing previously compared pairs can help avoid wasteful or repetitive computation. Another lookup table can also store the results of repeated exponential calculations (i.e., $(1 + \alpha)^n$ for all encountered values of n).

7 Evaluation

In our evaluation, we strive to answer the following questions by applying our method to several real and synthetic datasets of varying sizes:

1. *Scalability* How efficient is our hashing-based approach and how does it compare to baselines?
2. *Accuracy* How do our output graphs perform in real applications, such as classification of patient and healthy brains?
3. *Robustness* How do the scalability and accuracy of ABC-LSH change as parameters vary?

We ran all experiments and evaluation, written in single-threaded Python 3, on a single Ubuntu Linux server with 12 processors and 256 GB of RAM. For reproducibility, the code is available at https://github.com/tsafavi/hashing-based-network-discovery. We do not employ any extra optimizations (Sect. 6). As our methods achieve faster results than baselines *without* necessitating extra optimizations, we leave implementation of these optimizations for future work.

Table 2 All datasets used in our experiments and evaluation

Dataset	Description
COBRE	Resting-state fMRI from 72 patients with schizophrenia and 75 healthy controls. Each subject is associated with 1166 time series (brain regions) measured for around 100 timesteps
Penn	Resting-state fMRI from 519 subjects. Each subject's brain comprises 3789 regional time series measured for 110 timesteps
Synth-Penn	100 k time series either taken directly from a single brain in the Penn dataset or else randomly selected, phase-shifted versions of time series for the same brain
StarLightCurves	10 k phase-aligned time series of celestial object brightness values over time, each series of length 1024

7.1 Data

We used several datasets from different domains in our evaluation (Table 2), focusing on brain networks discovered from resting-state functional magnetic resonance imaging (fMRI).

Brain data In recent years, psychiatric and imaging neuroscience have shifted away from the study of segregated or localized brain functions toward a dynamic network perspective of the brain, where statistical dependencies and correlations between activity in brain regions can be modeled as a graph [15]. One well-known data collection procedure from which the brain's network organization may be modeled is resting-state fMRI, which tracks temporal and spatial oscillations of activity in neuronal ensembles [35]. Key objectives of resting-state fMRI are to elucidate the network mechanisms of mental disorders and to identify diagnostic biomarkers—objective, quantifiable characteristics that predict the presence of a disorder—from brain imaging scans [5]. Indeed, a fundamental hypothesis in the science of functional connectivity is that cognitive dysfunction can be illustrated and/or explained by a disturbed functional organization.

We used two publicly available datasets in this domain, COBRE [9] and Penn [37]. Both datasets were subject to a standard preprocessing pipeline, including linear detrending, or removal of low-frequency signal drift; removal of nuisance effects by regression; band-pass filtering, or rejection of frequencies out of a certain range; and censoring or removal of timesteps with high framewise motion.

Larger datasets To evaluate the scalability of ABC-LSH on larger datasets, we used a synthetic dataset with *more* time series and a real dataset with *longer* time series. Synth-Penn consists of 100,000 length-100 time series that were either taken directly from a single brain in the Penn dataset or else were randomly selected, phase-shifted versions of time series from the same brain. The average absolute correlation in Synth-Penn is $|r| = 0.22$. For the purpose of evaluating scalability as the number of time series grows, we generated graphs out of the first 1000, 2000, 5000, 10,000 and 20,000 series before using the full dataset.

We also used the StarLightCurves dataset, one of the largest time series datasets from the UCR Time Series archive, comprising 9236 phase-aligned time series that encode celestial body brightness values over 1024 timesteps [11]. The average absolute pairwise correlation in StarLightCurves is $|r| = 0.577$. Again, for the purpose of evaluating scalability as the number of time series grows, we constructed graphs out of the first 1000, 2000 and 5000 time series as well as the full dataset.

Table 3 Baselines to which we compare ABC and ABC-LSH

Baseline	Similarity or distance measure
Pairwise correlation	$r = \frac{\sum_i (x_i - \mu_{\mathbf{x}})(y_i - \mu_{\mathbf{y}})}{\sqrt{\sum_i (x_i - \mu_{\mathbf{x}})^2}\sqrt{\sum_i (y_i - \mu_{\mathbf{y}})^2}}$
Pairwise Euclidean distance	$d_{ED}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$
Window-LSH	$d_W(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \delta_{x[i:i+k-1] \neq y[i:i+k-1]}$

7.2 Task setup

Baselines In our **scalability** and **robustness** evaluations, we compared ABC-LSH to the standard in network discovery, pairwise absolute-valued Pearson's correlation. For our **task-based evaluation of accuracy**, we compared both pairwise and hashing-based network discovery using normalized ABC to three baselines (Table 3):

1. *Pearson's correlation* The standard in neuroscience.
2. *Euclidean distance* Although we show that Euclidean distance can fail to capture consecutiveness (Sect. 4), we use it as a baseline due to its simplicity, versatility, and popularity. To convert $d_{ED}(\mathbf{x}, \mathbf{y})$ to a similarity $s_{ED}(\mathbf{x}, \mathbf{y}) \in [0, 1]$, we compute the normalized Euclidean distance $d'_{ED}(\mathbf{x}, \mathbf{y})$ and then take $s_{ED}(\mathbf{x}, \mathbf{y}) = e^{-d'_{ED}(\mathbf{x}, \mathbf{y})}$ as the similarity score following the method described by Jäkel et al. [20].
3. *Window-LSH* We introduce a new metric, window distance, for use with the previously proposed window hashing family (Sect. 5.2). Described in more detail below, the window distance is somewhat more comparable to ABC, as it also emphasizes consecutiveness.

As all existing time series distance and similarity measures use pointwise comparisons, our proposed third baseline uses the window hashing family $\mathbf{F}_W$ with another consecutive-based distance measure. Window distance, also operating on binary sequences, is a simple extension of the Hamming distance $d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \delta_{x_i \neq y_i}$. We extend the Hamming distance to count the number of length-k *windows* (subsequences) at index i that do not match exactly between two binary sequences: $d_W(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \delta_{x[i:i+k-1] \neq y[i:i+k-1]}$. The window distance can be seen as the Hamming distance between vectors of $n - k + 1$ components, each component a length-k window in the original sequence. In the case of $k = 1$, the window distance reduces to the Hamming distance.

By extension of Hamming distance, it can be shown that the window distance is a metric ("Appendix A.2") and moreover has a window sampling LSH family ("Appendix A.4"). The hashing algorithm for Window-LSH is exactly the same as ABC-LSH. The only difference is that Window-LSH uses window distance, rather than ABC distance, and thus its LSH family properties p_1 and p_2 differ ("Appendix A.4").

Parameters To avoid an arbitrary edge-weight threshold θ for networks discovered with all-pairs correlation, we performed cross-validation following the task setup in Sect. 7.4 for $\theta = 0.15i$ where $i \in [1, 6]$. We found that $\theta = 0.6$ best balances classification accuracy with the runtime of producing computationally expensive per-graph feature values (i.e., average path length), which can be prohibitively slow to compute on very dense graphs.

We set the parameters for LSH with the goal of avoiding false negatives. Beyond facilitating comprehensive nearest-neighbor search and more accurate network discovery, this allowed for a "worst-case" comparison of scalability with pairwise correlation, as avoiding false

negatives requires more hashing and thus more computation. For the (real and synthetic) brain data, in which the time series consist of around 100 steps, we set $d_1 = 10$ and $d_2 = 95$. To avoid false negatives, we set $b = 8$ (OR) and $r = 1$ (AND) such that we were guaranteed with a 99.99% probability that $d(\mathbf{x}, \mathbf{y}) \leq 10$ for any colliding pair $(\mathbf{x}, \mathbf{y})$. For the `StarLightCurves` dataset with length-1024 time series, we set $d_1 = 64$ and $d_2 = 960$. We chose $r = 6$ and $b = 2$ for false positive and negative rates of less than 1%. Through cross-validation, we set k on the order of $\sqrt{n}$, the length of the time series: $k = 10$ for the brain data and $k = 64$ for `StarLightCurves`.

7.3 Question 1: scalability

Brain data On average, ABC-LSH was 9× faster on the `COBRE` dataset and 6.6× faster on the `Penn` dataset (Table 4). Since LSH is randomized, we averaged runtimes for LSH over three trials. We find that though the brain data are relatively small, meaning that all-pairs comparison is not particularly costly, graph generation with LSH was still an order of magnitude faster. For example, generating all brain networks in the `Penn` data took over 36 h (on average, 5 min/graph) with pairwise correlation, whereas generating the same brain networks with LSH took around 5.5 h (on average, 38 s/graph).

Larger datasets The scalability differences grew more pronounced as the number of time series increased. Pairwise correlation with $N = 20{,}000$ for a *single* brain took over 3 h, whereas ABC-LSH took on average 13 min (Fig. 7). Moreover, pairwise correlation with $N = 100{,}000$ ran out of memory, whereas ABC-LSH took less than 2 h.

While ABC-LSH was faster than pairwise correlation on the `StarLightCurves` dataset, it was slower overall than on the brain data. This result is not surprising, as the number of LSH comparisons depends heavily on the nature of the data and the average similarity of points in the dataset. The average absolute correlation between time series in `StarLightCurves` is more than twice that of `Synth-Penn`. As a result, more time series hashed to the same buckets. However, ABC-LSH still outperformed pairwise correlation on `StarLightCurves` at 2–4× faster (Fig. 7).

Table 4 LSH speedup on brain data (s)

Dataset	Corr.	LSH	LSH speedup
`COBRE`	3969	441	9×
`Penn`	131,307	19,722	6.6×

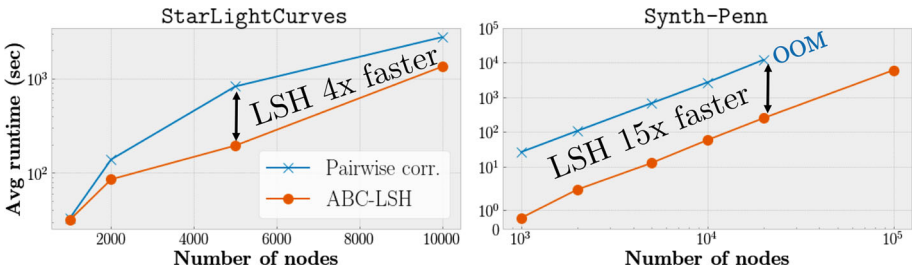


Fig. 7 Scalability of pairwise correlation versus ABC-LSH (runtime shown in log scale). As the number of nodes increases, ABC-LSH is up to 15× faster than the baseline, pairwise correlation. "OOM" denotes that pairwise correlation ran out of memory for $N = 100{,}000$

7.4 Question 2: accuracy

Our goal is not only to scale network discovery, but to also construct graphs that are as useful to practitioners as the standard networks built with pairwise correlation. Here we focused on the brain data, as the neuroscience and neuroimaging communities are rich with research and data on functional brain networks.

Brain network structure Two network-theoretical properties often studied in functional connectivity are the graph clustering coefficients and average path lengths [12], which are hypothesized to encode physical meaning on regional brain communication.

As we previously observed a strong linear relationship between correlation scores and ABC scores (Sect. 4.3), we hypothesized that the weighted networks discovered by correlation and ABC would also have similar structures. To confirm this, we computed the global clustering coefficient and average path length on all discovered networks in the `COBRE` and `Penn` datasets using pairwise correlation, pairwise ABC ($\theta = 0.6$), and ABC-LSH. Indeed, we found that the averages for both pairwise ABC and ABC-LSH are good approximations of pairwise correlation (Fig. 8).

Brain health classification Since we do not have ground-truth networks for our datasets, as is often the case in network discovery tasks, we evaluate the quality and distinguishability of the constructed brain networks through task-based evaluation. We classify the `COBRE` brain networks, which have associated control/schizophrenic labels, discovered by each of our baselines and the ABC variants. To represent the discovered networks, we used feature vectors of network properties commonly computed in functional connectivity: [density, average weighted degree, average clustering coefficient, modularity, average path length]^T.

In our evaluations, we used two classifiers for thoroughness, the first an SVM with an RBF kernel and the second a logistic regression classifier. We performed grid search over the classifier parameters to identify the best classification settings and used tenfold cross-validation to compute average performance metrics. We found that with both classifiers, the best performers were pairwise correlation and the ABC variants (Table 5).

Using the SVM classifier, pairwise ABC performed the best at 68% accuracy, 6% higher than the next-best ABC-LSH and 7% higher than pairwise correlation. With the logistic regression classifier, pairwise correlation performed the best at 68 percent accuracy, followed by ABC-LSH and pairwise ABC at 66 and 65% accuracy, respectively. As shown in

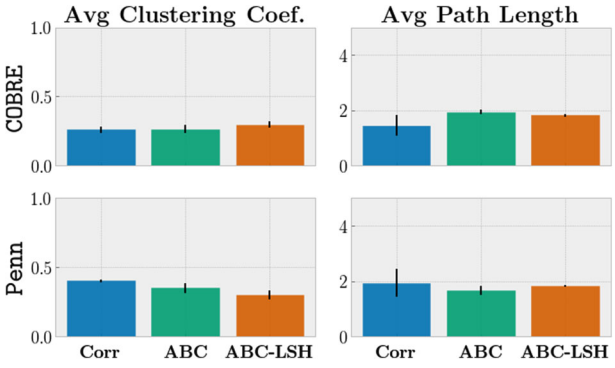


Fig. 8 Comparison of averaged brain network structural properties between correlation and the ABC variants across all `COBRE` and `Penn` subjects

Table 5 Best classification scores per network discovery method (SVM classifier/logistic regression classifier)

Method	Accuracy	Precision	Recall
Pairwise correlation	.61/**.68**	.59/**.66**	**.72/.80**
Pairwise Euclidean	.49/.57	.48/.57	.41/.59
Window-LSH	.52/.57	.53/.55	.45/.73
Pairwise ABC	**.68**/.65	**.71/.63**	.61/**.76**
ABC-LSH	**.62/.66**	**.66/.63**	**.63/.76**

Top 2 scores per classifier and method in bold. The ABC variants and pairwise correlation perform the best by far

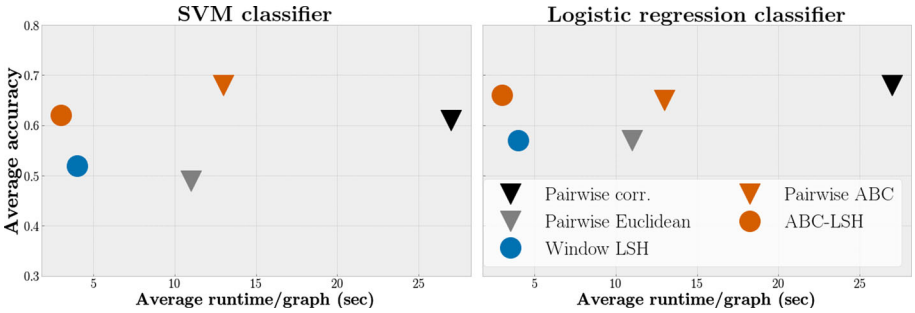


Fig. 9 Runtime versus classification accuracy. High accuracy and low runtime (top left quadrant) is best. In terms of accuracy, pairwise ABC and ABC-LSH perform comparably to, or better than, the established baseline pairwise correlation. ABC-LSH is the fastest. The other baselines, Euclidean distance and Window-LSH, perform poorly in terms of accuracy

Fig. 9, our ABC variants achieve *comparable or higher accuracy* than baselines. In particular, ABC-LSH performs in the same range as pairwise correlation with both classifiers, at 1% higher accuracy with the SVM and 2 percent lower accuracy with the logistic regression classifier. Furthermore, ABC-LSH is the fastest of all network discovery methods, and is *significantly faster* than the baseline pairwise correlation, as discussed in more detail in Sect. 7.3.

The other two baselines, Euclidean distance and Window-LSH, are faster than pairwise correlation and pairwise ABC, but lag far behind in accuracy. With the SVM, the Euclidean distance-based networks achieved 19% lower accuracy than pairwise ABC. With the logistic regression classifier, Euclidean distance achieved 9% lower accuracy than ABC-LSH. These results confirm the utility of quantifying time-consecutive similarity, whether through pairwise comparisons or, better yet, hashing. Even with approximate similarity search via LSH, identifying consecutively similar fluctuations among pairs of time series can lead to results better than baselines at a fraction of the computational cost.

Our results also indicate that quantifying *variable-length* consecutiveness in similarity scoring is much more flexible and accurate than counting the number of exactly matching consecutive intervals of fixed length. Accordingly, pairwise ABC and ABC-LSH performed upwards of 10% better than Window-LSH. The latter assigns low similarity scores to most pairs of binary sequences due to the relatively low likelihood of many exactly matching windows.

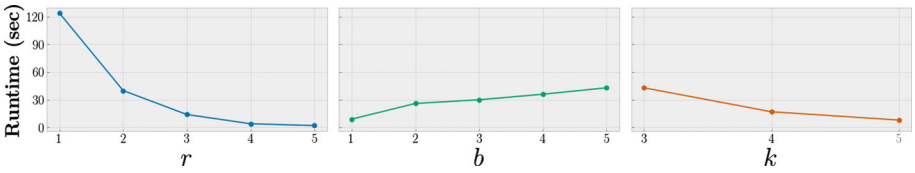


Fig. 10 ABC-LSH parameters and scalability. Varying the number of windows per signature r, number of hash tables b, and length of the sampled window k affects the number of collisions, which in turn affects computational runtime

7.5 Question 3: robustness

Finally, we investigate the effects of changing ABC-LSH parameters on scalability and network structure. To study these effects, we generated a single brain network from the COBRE dataset with a variety of LSH parameter settings, holding $\alpha = 10^{-4}$:

1. *AND construction* We varied the number of windows per signature $r \in [1, 5]$, holding $b = 4$ and $k = 3$.
2. *OR construction* We varied the number of hash tables $b \in [1, 5]$, holding $r = 2$ and $k = 3$.
3. *Window length* We varied the sampled subsequence length $k \in [3, 5]$, holding $r = 2$ and $b = 4$.

Scalability The number of sequences compared by ABC-LSH depends on parameter choices, which in turn affects false positive and negative rates. As expected, increasing r and/or k corresponded with a decrease in runtime, whereas increasing b corresponded with an increase in runtime (Fig. 10):

1. *AND construction* By increasing r, the length of each hash signature increases. The longer each hash signature, the less likely that hash signatures will match exactly, so runtime decreases due to fewer collisions.
2. *OR construction* By increasing b, the number of hash tables increases. This results in more opportunities for collisions, so runtime increases.
3. *Window length* By increasing k, the more unlikely that two time series windows will match exactly. Thus, runtime decreases due to fewer collisions, as is the case with increasing r.

Network structure As discussed previously, two properties often studied in functional networks are the clustering coefficient and average path length. We found relatively stable results in computing these properties across the specified parameter ranges, indicating that the discovered network structure is robust to ABC-LSH parameter changes (Fig. 11). With small fluctuations, the average path lengths stayed short (< 2.5) and the average clustering coefficients hovered around 0.4 to 0.6.

8 Discussion

In our experiments, we demonstrate that ABC-LSH is fast, accurate, and robust. Here, we address further questions that a reader may have about ABC-LSH.
Question 1 Does ABC-LSH address time series lag?
Answer Beyond the theoretical requirements of LSH (Sect. 2.3) and motivation (Sect. 4) that disqualify nonlinear time series alignment in our proposed methods, many domains interested

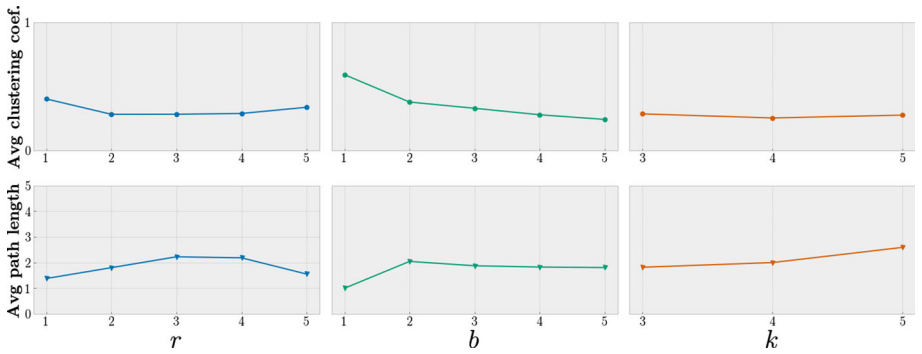


Fig. 11 ABC-LSH parameters and discovered network structure. The network properties remain relatively robust varying r, b, and k

in network discovery actually *seek* to identify linearly aligned correlations or associations between time series. For example, in neuroscience, the goal of studying brain networks is to discover which regions of the brain "activate" at the same time. As this is our main application-based motivation, we do not address time series lag in this work.

Question 2 What about network discovery via graphical models?

Answer Network inference methods using graphical models usually rely on different assumptions and are applied in different domains or tasks. Assuming some distribution of edges in the hidden network, these maximum likelihood-based methods aim to learn models that best fit both the empirical observations and the distributional assumptions. As discussed in our overview of related work in network discovery (Sect. 2.1), we choose not make such assumptions. Furthermore, as Brugere et al. [7] note, functional brain networks in the neuroscience literature are "almost exclusively" direct interaction networks based on thresholded pairwise similarity. As such, we followed the domain standard when designing our proposed methods.

Question 3 Why not use existing LSH families?

Answer To the best of our knowledge, no currently existing metric or LSH family quantifies time ordering or consecutiveness (Sect. 4). In short, our aim with ABC and ABC-LSH is to retain sequential information even in the hashing process, which is by nature approximate. Furthermore, pairwise Euclidean distance-based network discovery does not empirically perform well compared to pairwise correlation or our ABC variants (Sect. 7.4). Existing LSH families related to Euclidean distance [28] would likely not perform better, since they approximate the exact pairwise computations.

9 Conclusion

In this work, we motivate the problem of efficient network discovery, drawing from the vibrant research area of functional connectivity. To scale the existing quadratic-or-higher methods, we propose time-aware hashing. ABC-LSH is a 3-step approach that approximates time series, hashes them via window sampling, and builds a network using the results of hashing. In doing so, we introduce a novel sequence similarity measure, ABC, and a corresponding window sampling LSH family, ABC-LSH.

Using several datasets, we show that ABC-LSH is robust, discovers networks up to 15× faster than baselines (when the baselines do not run out of memory), and maintains

or improves accuracy in task-based evaluation. Furthermore, ABC-LSH is modular and thus generalizable to other sequence similarity and hashing tasks. Our work opens up many possibilities for future study at the intersection of networks, hashing, and time series. This in turn will impact a variety of domains as we continually seek new knowledge from data at an ever-increasing scale.

Acknowledgements We thank the anonymous reviewers for their useful comments and suggestions. This material is based upon work supported by the National Science Foundation under Grant No. IIS 1743088, Trove. AI, Google, and the University of Michigan. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Appendix A: Window distance measure

Given that most existing similarity measures on time series capture pointwise similarity, we introduce a new baseline approach emphasizing consecutiveness, window distance. The window distance is a simple extension of the Hamming distance. Hamming distance, which has been shown to be a metric, is computed pointwise. We extend it here to work with sequence *windows*, or subsequences.

Appendix A.1: Defining a mapping

Given a binary sequence $\mathbf{x} \in \{0, 1\}^n$, we can segment $\mathbf{x}$ into contiguous overlapping subsequences of length $k \in [1, n-k+1]$ and define a correspondence between $\mathbf{x}$ and its "window mapping" $w_k(\mathbf{x})$:

Definition 8 (*Window mapped series of x*) Given a binarized time series $\mathbf{x} \in \{0, 1\}^n$ and an integer $k \in [1, n-k+1]$, $\mathbf{x}$'s window mapping is

$$\begin{aligned} w_k(\mathbf{x}) &= [w_k(x_1), w_k(x_2), \ldots, w_k(x_{n-k+1})] \\ &= [(x_1, \ldots, x_k), (x_2, \ldots, x_{k+1}), \ldots, (x_{n-k+1}, \ldots, x_n)] \end{aligned} \tag{3}$$

The number of individual subsequences of $\mathbf{x}$ in $w_k(\mathbf{x})$ will be $n-k+1$, and each subsequence itself will be of length k, so the total length of $w_k(\mathbf{x})$ is $k(n-k+1)$.

Example 4 Given $\mathbf{x} = \mathbf{10110101}$, $\mathbf{y} = \mathbf{10101101}$, and $k = 3$, the respective window mappings are $w_3(\mathbf{x}) = [101, 011, 110, 101, 010, 101]$ and $w_3(\mathbf{y}) = [101, 010, 101, 011, 110, 101]$.

Appendix A.2: Window distance

Given two binary sequences $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ and their corresponding window mappings $w_k(\mathbf{x})$, $w_k(\mathbf{y}) \in \{0, 1\}^{k \times (n-k+1)}$, we can simply compute the Hamming distance between the mappings. In other words, we count the number of length-k windows at index i that do not match exactly between the sequences.

Definition 9 (*Window distance measure*) Given two binarized time series $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ and an integer $k \in [1, n-k+1]$, the window distance between the series is the number of

components between the respective window mappings $w_k(\mathbf{x})$ and $w_k(\mathbf{y})$ that do not match exactly:

$$d_W(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \delta_{x[i:i+k-1] \neq y[i:i+k-1]} \tag{4}$$

In essence, we are computing the Hamming distance between vectors with $n - k + 1$ components, each component a sequence encoding time-consecutive windows of length k in the original sequences.

Example 5 Given $\mathbf{x} = \mathbf{10110101}$, $\mathbf{y} = \mathbf{10101101}$, and $k = 3$, the distance $d(\mathbf{x}, \mathbf{y})$ is 4, as there are four windows that do not agree exactly:

$$w_3(\mathbf{x}) = [101, \mathbf{011}, \mathbf{110}, \mathbf{101}, \mathbf{010}, 101]$$
$$w_3(\mathbf{y}) = [101, \mathbf{010}, \mathbf{101}, \mathbf{011}, \mathbf{110}, 101]$$

Window similarity We can turn the window distance into a normalized similarity score between 0 and 1, which is useful for creating weighted similarity graphs, by subtracting the normalized observed distance between $\mathbf{x}$ and $\mathbf{y}$ from 1. The normalized distance is found by dividing by $n - k + 1$. In the example above, the similarity between $\mathbf{x}$ and $\mathbf{y}$ is $1 - \frac{4}{6} = \frac{1}{3}$.

Definition 10 (*Window similarity measure*) Given two binarized time series $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ and an integer $k \in [1, n - k + 1]$, the window similarity between the series is

$$s_W(\mathbf{x}, \mathbf{y}) = 1 - \frac{d_W(\mathbf{x}, \mathbf{y})}{n - k + 1} \tag{5}$$

Appendix A.3: Metric criteria

The window distance satisfies the criteria for a metric in the same way that the Hamming distance does.

1. *Identity* $d_W(\mathbf{x}, \mathbf{y}) = 0 \leftrightarrow \mathbf{x} = \mathbf{y}$. If $\mathbf{x}$ and $\mathbf{y}$ are the same, all of their windows will agree. Likewise, if all of the windows are the same, $\mathbf{x}$ and $\mathbf{y}$ will be the same.
2. *Non-negativity* $d_W(\mathbf{x}, \mathbf{y}) \geq 0$. The smallest number of windows that can disagree between two equal-length bit sequences $\mathbf{x}$ and $\mathbf{y}$ is 0.
3. *Symmetry* $d_W(\mathbf{x}, \mathbf{y}) = d_W(\mathbf{y}, \mathbf{x})$. The distance does not depend on which sequence is considered first.
4. *Triangle inequality* $d_W(\mathbf{x}, \mathbf{y}) \leq d_W(\mathbf{x}, \mathbf{z}) + d_W(\mathbf{z}, \mathbf{y})$. This measure is a version of Hamming distance, which has been shown to satisfy the triangle inequality [28]. Essentially, if a is the number of components that disagree between $w_k(\mathbf{x})$ and $w_k(\mathbf{z})$, and b is the number of windows that disagree between $w_k(\mathbf{z})$ and $w_k(\mathbf{y})$, the number of windows that disagree between $w_k(\mathbf{x})$ and $w_k(\mathbf{y})$ cannot be more than $a + b$.

Appendix A.4: Window-LSH

Our proposed window sampling LSH family (Sect. 5.2) using the window metric rather than ABC distance is $(d_1, d_2, 1 - \frac{d_1}{n-k+1}, 1 - \frac{d_2}{n-k+1})$-sensitive. As was the case with Hamming distance, we normalize the distances d_1 and d_2 by dividing by the maximum distance, $n-k+1$, and then subtract from 1 to turn the distance into a probability.

Appendix B: Metric proof of ABC

Here we show that ABC distance satisfies the metric properties and is thus eligible for LSH.

Appendix B.1: Properties of agreeing runs

We first study the relationship between p, the number of agreeing runs between $\mathbf{x}$ and $\mathbf{y}$, and the maximum value of $k_1 + \cdots + k_p$, the lengths of the p agreeing runs. The maximum sum of all k_i decreases linearly as p increases.

Lemma 1 (Maximum sum of lengths of p runs $k_1, \ldots, k_p$) *Given $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ with p agreeing runs, each of length k_i, the maximum sum of the lengths of the p runs $k_1, \ldots, k_p$ follows a linearly decreasing relationship, as $\sum_{i=1}^{p} k_i = n - p + 1$.*

Proof We show that the maximum value of $\sum_1^p k_i$ must decrease as p increases.

1. If $p = 1$, $k_1 \leq n$. In other words, if there is a single matching run between $\mathbf{x}$ and $\mathbf{y}$, the length of the matching run can be anywhere between 1 bit to n bits.
2. If $p = 2$, $k_1 + k_2 \leq n - 1$. Proof by contradiction: assume $k_1 + k_2 = n$. Then there is a matching run of k_1 bits between $\mathbf{x}$ and $\mathbf{y}$, and the remaining $n - k_1 = k_2$ bits of $\mathbf{x}$ and $\mathbf{y}$ are also a matching run, which means that the two matching runs are consecutive, making them one long run. This means $p = 1$, which contradicts the initial assumption that $p = 2$. In other words, $k_1 + k_2 \leq n - 1$ because there must be at least one bit separating the run of the length k_1 and the run of length k_2.
3. If $p = 3$, $k_1 + k_2 + k_3 \leq n - 2$. This follows from the rule above, since if $k_1 + k_2 + k_3$ were $n - 1$, one pair of runs would have to be merged, making $p = 2$.
4. Following the observations above, $\sum_{i=1}^{p} k_i \leq n - p + 1$, with equality only when $\sum_{i=1}^{p} k_i$ is maximized. Thus, the maximum sum of all k_i follows an inverse linear relationship with p.

□

Lemma 2 (Relationship between p and $\sum_1^p k_i$) *Given $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ with p agreeing runs, each of length k_i, as p increases, $\sum_{i=1}^{p} k_i \leq n - p + 1$, as the number of bits that separate runs increases with p.*

Next, we investigate the values of $k_1, \ldots, k_p$ themselves for the "maximum similarity" $S_p(\mathbf{x}, \mathbf{y})$ given a value of p. First, we observe that for maximizing similarity given a p, the values k_i for the lengths of the p runs must be

$$\begin{aligned} k_1 = \cdots = k_{p-1} &= 1 \\ k_p &= n - 2p + 2 \end{aligned} \tag{6}$$

Intuitively, this observation makes sense because by making $p - 1$ agreeing runs as short as possible, the length of the p-th run is maximized, adding on the common ratio $(1 + \alpha)$ raised to the greatest exponent possible. Furthermore, note how for any value of p in the above, $\sum_{i=1}^{p} k_i = n - p + 1$ because $(p - 1) + n - 2p + 2 = n - p + 1$. This fits with our previous observations: since the similarity is a summation of positive terms, we want to maximize the number of runs and the sum of their lengths.

Lemma 3 (Maximum ABC similarity) *Given $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ with p agreeing runs, each of length k_i, $\boldsymbol{x}$ and $\boldsymbol{y}$ have maximum ABC similarity when they agree on (without loss of*

generality, the first) $p-1$ *runs of length* $k_1 = \cdots = k_{p-1} = 1$ *and one run of length* $k_p = n - 2p + 2$.

Proof To confirm the intuition, we prove by induction that $k_1 = \cdots = k_{p-1} = 1$ and $k_p = n - 2p + 2$ for maximum similarity given a p and $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$:

1. *Base case* $p = 1$. The similarity between $\mathbf{x}$ and $\mathbf{y}$ is maximized when $\mathbf{x}$ and $\mathbf{y}$ are exactly equal and the length of the single run is n. Thus, $k_p = k_1 = n = (1-1)+n-2(1)+2 = (p-1)+n-2p+2$.
2. *Inductive step* Assume for some p, the similarity between $\mathbf{x}$ and $\mathbf{y}$ is maximized when $k_1 = \cdots = k_{p-1} = 1$ and $k_p = n - 2p + 2$. We show that this implies that for some $p+1$, the similarity is maximized when $k_1 = \cdots = k_p = 1$ and $k_{p+1} = n-2(p+1)+2$, assuming the same constraint $p+1 \leq \frac{n}{2}$. Since by the inductive hypothesis the similarity between the first p runs is maximized when $k_1 = \cdots = k_{p-1} = 1$ and $k_p = n-2p+2$, the "best we can do" given that we have to add a new run is to take one bit out of the long run (the run of length k_p) to contribute to the $(p+1)$-th run, and remove another bit from the long run such that the $(p+1)$-th run and the run of length k_p are not consecutive. Thus we have $p = (p+1)-1$ runs of length 1, and a long run of length $n-2p = n-2(p+1)+2$. Thus, we have shown that the inductive hypothesis for some p implies that the hypothesis is true for $p+1$, so by induction we maximize similarity between $\mathbf{x}$ and $\mathbf{y}$ with $k_1 = \cdots = k_{p-1} = 1$ and $k_p = n - 2p + 2$.

□

Definition 11 (*Maximum ABC similarity given p*) Based on Theorem 3, the maximum ABC similarity $S(\mathbf{x}, \mathbf{y})_p$ between two binary time series $\mathbf{x}$ and $\mathbf{y}$ with p agreeing runs is

$$S(\mathbf{x}, \mathbf{y})_p = (p-1) + \frac{(1+\alpha)^{n-2p+2} - 1}{\alpha} \tag{7}$$

Likewise, the minimum ABC distance between two binary sequences $\mathbf{x}$ and $\mathbf{y}$ given a p is

$$\min_p d(\mathbf{x}, \mathbf{y}) = \frac{(1+\alpha)^n - (1+\alpha)^{n-2p+2}}{\alpha} - p + 1 \tag{8}$$

Appendix B.2: Proving the triangle inequality for ABC distance

Our main result that enables scalable network discovery is that ABC distance is a metric satisfying the triangle inequality.

Proof We begin with the **base case** where $n = 1$: $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are single bits.

1. $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are the same: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}, \mathbf{z}) = d(\mathbf{z}, \mathbf{y}) = 0$. $0 \leq 0$.
2. $\mathbf{x}$ and $\mathbf{y}$ are the same, $\mathbf{z}$ is different: $d(\mathbf{x}, \mathbf{y}) = 0$, $d(\mathbf{x}, \mathbf{z}) = 1$, and $d(\mathbf{z}, \mathbf{y}) = 1$. $0 \leq 2$.
3. $\mathbf{x}$ and $\mathbf{z}$ are the same, $\mathbf{y}$ is different: $d(\mathbf{x}, \mathbf{y}) = 1$, $d(\mathbf{x}, \mathbf{z}) = 0$, and $d(\mathbf{z}, \mathbf{y}) = 1$. $1 \leq 1$.
4. $\mathbf{y}$ and $\mathbf{z}$ are the same, $\mathbf{x}$ is different: $d(\mathbf{x}, \mathbf{y}) = 1$, $d(\mathbf{x}, \mathbf{z}) = 1$, and $d(\mathbf{z}, \mathbf{y}) = 0$. $1 \leq 1$.

Next, we move to the **inductive step**. Assume that $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ for some value of $n > 1$. We show that this implies that the inequality holds for binarized series of length $n+1$, which are constructed by adding a single bit to the end of each of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ to create $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, respectively.

Setup We begin with preliminaries. First, we denote the distance between $\mathbf{x}$ and $\mathbf{y} \in \{0,1\}^n$ as $d(\mathbf{x}, \mathbf{y})$:

$$d(\mathbf{x}, \mathbf{y}) = \frac{(1+\alpha)^n - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} + p - 1}{\alpha}$$

Next, we denote the distance between $\mathbf{x}'$ and $\mathbf{y}' \in \{0,1\}^{n+1}$ as $d'(\mathbf{x}', \mathbf{y}')$:

$$d'(\mathbf{x}', \mathbf{y}') = \frac{(1+\alpha)^{n+1} - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_{p'}} + p' - 1}{\alpha}$$

Here, p' can either be p or $p+1$: $p' = p$ in the case that the $n+1$-th bit either appends onto an existing run between $\mathbf{x}$ and $\mathbf{y}$, or else disagrees between the two sequences, and $p' = p+1$ in the case that the $n+1$-th bit creates a new run of length 1 between $\mathbf{x}$ and $\mathbf{y}$.

We now examine how distance between $\mathbf{x}$ and $\mathbf{y}$ changes by adding a single bit to the end of $\mathbf{x}$ and $\mathbf{y}$: in other words, moving from n to $n+1$. We denote this change in distance $\Delta_{(i)}$ for $i = 1, 2$, or 3.

(1) Case 1: the $n+1$-th bits of $\mathbf{x}$ and $\mathbf{y}$ agree, creating a new run of length one between the sequences. Here $p' = p+1$, so $k_{p'} = 1$ and $(1+\alpha)^{k_{p'}} = (1+\alpha)$.

$$\begin{aligned}
\Delta_{(1)} &= d'(\mathbf{x}', \mathbf{y}') - d(\mathbf{x}, \mathbf{y}) \\
&= \frac{(1+\alpha)^{n+1} - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} - (1+\alpha) + (p+1) - 1}{\alpha} \\
&\quad - \frac{(1+\alpha)^n - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} + p - 1}{\alpha} \\
&= \frac{(1+\alpha)^{n+1} - (1+\alpha)^n - (1+\alpha) - 1}{\alpha} \\
&= \frac{(1+\alpha-1)(1+\alpha)^n - \alpha}{\alpha} \\
&= (1+\alpha)^n - 1
\end{aligned}$$

Intuitively this result means that the maximum similarity $S(\mathbf{x}, \mathbf{y})$ increases by $(1+\alpha)^n$, and from this we subtract a new agreeing run of length 1. In other words, we subtract $(1+\alpha)^0$ from the new maximum similarity since the exponent of a new run always begins at 0. Thus, overall the distance changes by $(1+\alpha)^n - (1+\alpha)^0 = (1+\alpha)^n - 1$.

(2) Case 2: the $n+1$-th bits of $\mathbf{x}$ and $\mathbf{y}$ agree, adding or appending onto an existing run of length k_p between the sequences. Here $p' = p$ and $k_{p'} = k_p + 1$, so $(1+\alpha)^{k_{p'}} = (1+\alpha)^{k_p+1}$.

$$\begin{aligned}
\Delta_{(2)} &= d'(\mathbf{x}', \mathbf{y}') - d(\mathbf{x}, \mathbf{y}) \\
&= \frac{(1+\alpha)^{n+1} - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p+1} + p - 1}{\alpha} \\
&\quad - \frac{(1+\alpha)^n - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} + p - 1}{\alpha} \\
&= \frac{(1+\alpha)^{n+1} - (1+\alpha)^n - (1+\alpha)^{k_p+1} - (1+\alpha)^{k_p}}{\alpha} \\
&= \frac{(1+\alpha-1)(1+\alpha)^n - (1+\alpha-1)(1+\alpha)^{k_p}}{\alpha} \\
&= (1+\alpha)^n - (1+\alpha)^{k_p}
\end{aligned}$$

Table 6 Enumeration of possible cases for the $n+1$-th bit in $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$

Case	Possible?	Explanation
ddd	✘	There are only two possibilities for the $n+1$-th bit, since we are working with binarized series. Since there are three series $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, by the pigeonhole principle at least two of the series must agree in the $n+1$-th bit
daa	✘	If $\mathbf{x}'$ and $\mathbf{z}'$ agree in the $n+1$-th bit, and $\mathbf{z}'$ and $\mathbf{y}'$ agree in the $n+1$-th bit, then $\mathbf{x}'$ and $\mathbf{y}'$ cannot disagree in the $n+1$-th bit
dan	✘	Same explanation as above (case **daa**)
dna	✘	Same explanation as above
dnn	✘	Same explanation as above
ada	✘	If $\mathbf{x}'$ and $\mathbf{y}'$ agree in the $n+1$-th bit, and $\mathbf{z}'$ and $\mathbf{y}'$ agree in the $n+1$-th bit, then $\mathbf{x}'$ and $\mathbf{z}'$ cannot disagree in the $n+1$-th bit
adn	✘	Same explanation as above (case **ada**)
ndn	✘	Same explanation as above
nda	✘	Same explanation as above
and	✘	If $\mathbf{x}'$ and $\mathbf{y}'$ agree in the $n+1$-th bit, and $\mathbf{x}'$ and $\mathbf{z}'$ agree in the $n+1$-th bit, then $\mathbf{z}'$ and $\mathbf{y}'$ cannot disagree in the $n+1$-th bit
aad	✘	Same explanation as above (case **and**)
nad	✘	Same explanation as above
nnd	✘	Same explanation as above
aan	✘	If the $n+1$-th bit of $\mathbf{x}'$ and $\mathbf{y}'$ appends to an existing run, and the $n+1$-th bit of $\mathbf{x}'$ and $\mathbf{z}'$ appends to an existing run, then there must be an existing run between $\mathbf{z}$ and $\mathbf{y}$, so the $n+1$-th bit cannot start a new run
naa	✘	If the $n+1$-th bit of $\mathbf{x}'$ and $\mathbf{z}'$ appends to an existing run, and the $n+1$-th bit of $\mathbf{z}'$ and $\mathbf{y}'$ appends to an existing run, then there must be an existing run between $\mathbf{x}$ and $\mathbf{y}$, so the $n+1$-th bit cannot start a new run
ana	✘	If the $n+1$-th bit of $\mathbf{x}'$ and $\mathbf{y}'$ appends to an existing run, and the $n+1$-th bit of $\mathbf{z}'$ and $\mathbf{y}'$ appends to an existing run, then there must be an existing run between $\mathbf{x}$ and $\mathbf{z}$, so the $n+1$-th bit cannot start a new run
nnn	✘	If the $n+1$-th bit starts a new run between all pairs, then all pairs must disagree in the n-th bit, which is not possible by case **ddd**
dda	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between $\mathbf{x}$ and $\mathbf{z}$. The distance between $\mathbf{z}$ and $\mathbf{y}$ changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$ where k_p is the length of the last run between $\mathbf{z}$ and $\mathbf{y}$. We have $(1+\alpha)^n \leq (1+\alpha)^n + (1+\alpha)^n - (1+\alpha)^{k_p}$, or $0 \leq (1+\alpha)^n - (1+\alpha)^{k_p}$. Since $k_p \leq n$, the inequality holds
dad	✔	Symmetric with above (case **dda**), with the Δ's between ($\mathbf{x}$, $\mathbf{z}$) and ($\mathbf{z}$, $\mathbf{y}$) flipped
ddn	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between $\mathbf{x}$ and $\mathbf{z}$. The distance between $\mathbf{z}$ and $\mathbf{y}$ changes by $\Delta_{(1)} = (1+\alpha)^n - 1$. We have $(1+\alpha)^n \leq (1+\alpha)^n + (1+\alpha)^n - 1$, or $0 \leq (1+\alpha)^n - 1$. Since $n > 0$, the inequality holds
dnd	✔	Symmetric with above (case **ddn**)

Table 6 continued

Case	Possible?	Explanation
add	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$ where k_p is the length of the last run between $\mathbf{x}$ and $\mathbf{y}$. The distance between $\mathbf{x}$ and $\mathbf{z}$ changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between $\mathbf{z}$ and $\mathbf{y}$. We have $(1+\alpha)^n - (1+\alpha)^{k_p} \leq (1+\alpha)^n + (1+\alpha)^n$, or $-(1+\alpha)^{k_p} \leq (1+\alpha)^n$. The right-hand side must be positive, so the inequality holds
nan	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(1)} = (1+\alpha)^n$, as does the distance between $\mathbf{z}$ and $\mathbf{y}$. The distance between $\mathbf{x}$ and $\mathbf{z}$ changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$, where k_p is the
nna	✔	Symmetric with above (case **nan**)
ndd	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(1)} = (1+\alpha)^n - 1$. The distance between $\mathbf{x}$ and $\mathbf{z}$, as well as the distance between $\mathbf{z}$ and $\mathbf{y}$, changes by $(1+\alpha)^n$. We have $(1+\alpha)^n - 1 \leq (1+\alpha)^n + (1+\alpha)^n$, or $0 \leq (1+\alpha)^n + 1$, so the inequality holds
ann	✔	The distance between $\mathbf{x}$ and $\mathbf{y}$ changes by $\Delta_{(3)} = (1+\alpha)^n - (1+\alpha)^{k_p}$, where k_p is the length of the last run between $\mathbf{x}$ and $\mathbf{y}$. The distance between $\mathbf{x}$ and $\mathbf{z}$ changes by $\Delta_{(1)} = (1+\alpha)^n - 1$, as does the distance between $\mathbf{z}$ and $\mathbf{y}$. We have $(1+\alpha)^n - (1+\alpha)^{k_p} \leq (1+\alpha)^n - 1 + (1+\alpha)^n - 1$, or equivalently $0 \leq (1+\alpha)^n + (1+\alpha)^{k_p} - 2$. $(1+\alpha)^n + (1+\alpha)^{k_p} \leq 2$, since each term is at least 1, so the inequality holds
aaa	✔	Let the length of last run between $\mathbf{x}$ and $\mathbf{y}$ be denoted k_{p1} and the length of the last run between $\mathbf{x}$ and $\mathbf{z}$ be noted k_{p2}. Then the distance between $\mathbf{x}$ and $\mathbf{y}$ increases by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_{p1}}$, and the distance between $\mathbf{x}$ and $\mathbf{z}$ increases by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_{p2}}$. Since the new bit has added onto an existing run between $\mathbf{x}$ and $\mathbf{y}$ of length k_{p1}, and an existing run between $\mathbf{x}$ and $\mathbf{z}$ of length k_{p2}, the longest the last run between $\mathbf{z}$ and $\mathbf{y}$ can be is $\min[k_{p1}, k_{p2}]$. For example, if the run between $\mathbf{x}$ and $\mathbf{y}$ is 3 bits and the run between $\mathbf{x}$ and $\mathbf{z}$ is 5 bits, the run between $\mathbf{y}$ and $\mathbf{z}$ can be at most 3 bits. Therefore, the distance between $\mathbf{z}$ and $\mathbf{y}$ changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{\min[k_{p1},k_{p2}]}$ If $k_{p1} < k_{p2}$, the inequality becomes $(1+\alpha)^n - (1+\alpha)^{k_{p1}} \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + (1+\alpha)^n - (1+\alpha)^{k_{p1}}$, or equivalently $0 \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}}$. Since $k_{p2} \leq n$, the inequality holds. If $k_{p1} > k_{p2}$, the inequality becomes $(1+\alpha)^n - (1+\alpha)^{k_{p1}} \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + (1+\alpha)^n - (1+\alpha)^{k_{p2}}$, or equivalently $0 \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + [(1+\alpha)^{k_{p1}} - (1+\alpha)^{k_{p2}}]$. The bracketed term must be greater than 0 because $k_{p1} > k_{p2}$. Since $(1+\alpha)^n - (1+\alpha)^{k_{p2}}$ because $k_{p2} \leq n$, we get a sum of two terms that are each greater than or equal to 0, so the inequality holds

(3) Case 3: the $n+1$-th bits of $\mathbf{x}$ and $\mathbf{y}$ disagree. Here $p' = p$ and $k_{p'} = k_p$.

$$\begin{aligned}\Delta_{(3)} &= d'(\mathbf{x}', \mathbf{y}') - d(\mathbf{x}, \mathbf{y}) \\ &= \frac{(1+\alpha)^{n+1} - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} + p - 1}{\alpha} \\ &\quad - \frac{(1+\alpha)^n - (1+\alpha)^{k_1} - \cdots - (1+\alpha)^{k_p} + p - 1}{\alpha}\end{aligned}$$

$$= \frac{(1+\alpha)^{n+1} - (1+\alpha)^n}{\alpha}$$
$$= \frac{(1+\alpha-1)(1+\alpha)^n}{\alpha}$$
$$= (1+\alpha)^n$$

Enumeration of cases With the preliminaries established, we list all cases that can occur when we add a single bit to $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ to obtain $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$.

Let $\mathbf{n}$ stand for "new run" (case **(1)** above with $\Delta_{(1)}$), $\mathbf{a}$ stand for "append to existing run" (case **(2)** with $\Delta_{(2)}$), and $\mathbf{d}$ stand for "disagree" (case **(3)** with $\Delta_{(3)}$). There are $3 \times 3 \times 3$ length-3 permutations with repetition of $\mathbf{n}$, $\mathbf{a}$, and $\mathbf{d}$ for three series $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, although not all are possible in practice: in fact, only 10 out of the 27 cases are feasible. In Table 6 we either explain why each case is impossible or else show that the triangle inequality holds.

We have shown by induction on n, the length of the binary subsequences compared, that the triangle inequality holds for the ABC distance measure because the amounts by which the distances change for any combination of new bits appended to $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ satisfy the triangle inequality. Thus the ABC distance satisfies the metric properties. □

References

1. Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688
2. Andoni A, Indyk P (2008) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. CACM 51(1):117–122
3. Ashkenazy Y, Ivanov PC, Havlin S, Peng C-K, Goldberger AL, Stanley HE (2001) Magnitude and sign correlations in heartbeat fluctuations. Phys Rev Lett 86(9):1900–1903
4. Balakrishnan N, Koutras M (2002) Runs and scans with applications. Wiley, Hoboken
5. Bassett D, Bullmore E (2009) Human brain networks in health and disease. Curr Opin Neurol 22(4):340–347
6. Bayardo RJ, Ma Y, Srikant R (2007) Scaling up all pairs similarity search. In: Proceedings of the 16th international conference on world wide web, pp 131–140
7. Brugere I, Gallagher B, Berger-Wolf TY (2018) Network structure inference, a survey: motivations, methods, and applications. ACM Comput Surv (CSUR) 51(2):24
8. Bullmore E, Sporns O (2009) Complex brain networks: graph theoretical analysis of structural and functional systems. Nat Rev Neurosci 10(3):186–198
9. Center for Biomedical Research Excellence (2012) http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html
10. Chaudhuri S, Ganti V, Kaushik R (2006) A primitive operator for similarity joins in data cleaning. In: Proceedings of the 22nd international conference on data engineering. ICDE '06
11. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/. Accessed 1 Jan 2017
12. Dai Z, He Y (2014) Disrupted structural and functional brain connectomes in mild cognitive impairment and Alzheimer's disease. Neurosci Bull 30(2):217–232
13. Davidson I, Gilpin S, Carmichael O, Walker P (2013) Network discovery via constrained tensor analysis of fmri data. In: KDD, pp 194–202
14. Dong W, Moses C, Li K (2011) Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th international conference on World wide web, ACM, pp 577–586
15. Friston KJ (2011) Functional and effective connectivity: a review. Brain Connect 1(1):13–36
16. Hallac D, Park Y, Boyd S, Leskovec J (2017) Network inference via the time-varying graphical lasso. In: 'KDD'
17. Heimann M, Lee W, Pan S, Chen K, Koutra D (2018) Hashalign: Hash-based alignment of multiple graphs. In: Advances in knowledge discovery and data mining—22nd Pacific-Asia conference, PAKDD 2018, Melbourne, VIC, Australia, June 3–6, 2018, Proceedings, Part III, pp 726–739
18. Iglesias F, Kastner W (2013) Analysis of similarity measures in times series clustering for the discovery of building energy patterns. Energies 6(2):579–597

19. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: 'STOC', pp 604–613
20. Jäkel F, Schlkopf B, Wichmann F (2008) Similarity, kernels, and the triangle inequality. J Math Psychol 52(5):297–303
21. Kale DC, Gong D, Che Z, Liu Y, Medioni G, Wetzel R, Ross P (2014) An examination of multivariate time series hashing with applications to health care. In: ICDM, pp 260–269
22. Keogh E, Pazzani M (1999) An indexing scheme for fast similarity search in large time series databases. In: SSDM, pp 56–67
23. Kim YB, Hemberg E, O'Reilly U-M (2016) Stratified locality-sensitive hashing for accelerated physiological time series retrieval. In: EMBC
24. Kim YB, O'Reilly U-M (2015) Large-scale physiological waveform retrieval via locality-sensitive hashing. In: EMBC, pp 5829–5833
25. Koutra D, Faloutsos C (2017) Individual and collective graph mining: principles, algorithms, and applications. In: Synthesis lectures on data mining and knowledge discovery. Morgan and Claypool Publishers
26. Koutra D, Shah N, Vogelstein JT, Gallagher B, Faloutsos C (2016) Deltacon: principled massive-graph similarity function with attribution. TKDD 10(3):28:1–28:43
27. Kuo C-T, Wang X, Walker P, Carmichael O, Ye J, Davidson I (2015) Unified and contrasting cuts in multiple graphs: application to medical imaging segmentation. In: KDD, pp 617–626
28. Leskovec J, Rajaraman A, Ullman JD (2014) Mining of massive datasets. Cambridge University Press, Cambridge
29. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: SIGMOD, pp 2–11
30. Liu Y, Safavi T, Dighe A, Koutra D (2018) Graph summarization methods and applications: a survey. ACM Comput Surv 51(3):62:1–62:34
31. Luo C, Shrivastava A (2016) SSH (Sketch, Shingle, and Hash) for indexing massive-scale time series. In: NIPS time series workshop
32. Martínez V, Berzal F, Cubero J-C (2016) A survey of link prediction in complex networks. ACM Comput Surv 49(4):69:1–69:33
33. Müller M (2007) Information retrieval for music and motion. Springer, New York
34. Onnela J-P, Kaski K, Kertsz J (2004) Clustering and information in correlation based financial networks. Eur Phys J B 38:353–362
35. Park H-J, Friston K (2013) Structural and functional brain networks: from connections to cognition. Science 342(6158):579–589
36. Ratanamahatana C, Keogh E, Bagnall AJ, Lonardi S (2005) A novel bit level time series representation with implication of similarity search and clustering. In: PAKDD, pp 771–777
37. Satterthwaite T, Elliott M, Ruparel K, Loughead J, Prabhakaran K, Calkins M, Hopson R, Jackson C, Keefe J, Riley M, Mentch F, Sleiman P, Verma R, Davatzikos C, Hakonarson H, Gur R, Gur R (2014) Neuroimaging of the Philadelphia neurodevelopmental cohort. Neuroimage 86:544–553
38. Scharwächter E, Geier F, Faber L, Müller E (2018) Low redundancy estimation of correlation matrices for time series using triangular bounds. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 458–470
39. Shah N, Koutra D, Jin L, Zou T, Gallagher B, Faloutsos C (2017) On summarizing large-scale dynamic graphs. IEEE Data Eng Bull 40(3):75–88
40. Shuman DI, Narang SK, Frossard P, Ortega A, Vandergheynst P (2013) The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process Mag 30(3):83–98
41. Tsitsulin A, Mottin D, Karras P, Bronstein AM, Müller E (2018) Netlsd: hearing the shape of a graph. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, KDD 2018, London, UK, August 19–23, 2018, pp 2347–2356
42. Yang S, Sun Q, Ji S, Wonka P, Davidson I, Ye J (2015) Structural graphical lasso for learning mouse brain connectivity. In: KDD, pp 1385–1394
43. Yeh C-CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E (2016) Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th international conference on data mining (ICDM), pp 1317–1322
44. Zhang Y-M, Huang K, Geng G, Liu C-L (2013) Fast kNN graph construction with locality sensitive hashing. In: ECML PKDD, pp 660–674

Tara Safavi is a graduate student in Computer Science and Engineering at the University of Michigan. Her research focuses on scalable graph-based data mining and machine learning. She is the recipient of an NSF Graduate Research Fellowship, a Google Women Techmakers scholarship, and a University of Michigan Dean's and Named PhD fellowship. She also has one best paper nomination and one patent pending.

Chandra Sripada is an Associate Professor with joint appointments in Psychiatry and Philosophy at the University of Michigan. His research examines agency, attention, and self-control from cross-disciplinary perspectives.

Danai Koutra is an Assistant Professor in Computer Science and Engineering at University of Michigan, where she leads the Graph Exploration and Mining at Scale (GEMS) Lab. She researches scalable graph mining methods for gaining insights into massive, messy, interconnected data, which are prevalent in the real world. Her interests include graph summarization, analysis of multi-source network data, similarity, matching, and anomaly detection. Her work has been applied to social, communication, and web networks, as well as to brain (functional) connectivity graphs. She won an ARO Young Investigator award and an Adobe Data Science Research Faculty Award in 2018, the 2016 ACM SIGKDD Dissertation award, and an honorable mention for the SCS Doctoral Dissertation Award (CMU). She has multiple papers in top data mining conferences, including 5 award-winning papers, and holds one "rate-1" patent and six (pending) patents on bipartite graph alignment. She is the Program Director of the SIAG on Data Mining and Analytics and an Associate Editor of ACM TKDD.