# Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches

**Miro Mannino, Azza Abouzied**
New York University Abu Dhabi, UAE
{miro.mannino, azza}@nyu.edu

## ABSTRACT

We present Qetch, a tool where users freely sketch patterns on a scale-less canvas to query time series data without specifying query length or amplitude. We study how humans sketch time series patterns — humans preserve visually salient perceptual features but often non-uniformly scale and locally distort a pattern — and we develop a novel matching algorithm that accounts for human sketching errors. Qetch enables the easy construction of complex and expressive queries with two key features: *regular expressions over sketches* and *relative positioning of sketches* to query multiple time-aligned series. Through user studies, we demonstrate the effectiveness of Qetch's different interaction features. We also demonstrate the effectiveness of Qetch's matching algorithm compared to popular algorithms on targeted, and exploratory query-by-sketch search tasks on a variety of data sets.

### ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

### Author Keywords

Time series querying by sketching; scale-less sketches; regular expressions

## INTRODUCTION

Our ability to describe complex objects with hand-drawn sketches and easily recognize them predates our ability to do so with language. Many search interfaces have capitalized on this ability, providing users with intuitive sketching interfaces: users sketch their object of interest, be it an image, a 3D-model or a chart pattern, and a matching algorithm finds similar objects in an image, model or time series database [11, 38, 29]. Generally, such *querying by sketching* systems assume that in a "well-engineered feature space, sketched objects resemble their real-world counterparts [10]". Yet, this fundamental assumption is often violated: "most humans are not faithful artists [10]." While simple stick-figures or cartoon-like sketches drastically differ from their real world counterparts, differences between a sketched chart pattern () and actual time series data (  ) are generally perceived as less drastic. Not surprisingly, the accuracy of many existing time series matching algorithms is often evaluated by how well they cluster similar time series patterns extracted from the data (, ) — and not how well they cluster data patterns with hand-drawn sketches [8]. Consequently, "good" matching algorithms may fail to produce good similarity rankings when "goodness" is assessed by humans [9]. Most query-by-sketching interfaces attempt to reconcile imperfect human sketches with how time series matching algorithms search for data through (a) *overlays:* users sketch directly above the pattern they are looking for to retrieve similar patterns in the data, (b) *shape restrictions:* instead of free-form sketches, users can only sketch sequences of straight lines, (c) *pre-sketching constraints:* users specify the temporal range they are interested in and the variance of amplitude they are willing to tolerate. In this work, we reconsider the design of query-by-sketching interfaces for time series data to allow users to *freely* sketch patterns on an empty, scale-less canvas. With Qetch, we adopt a top-down design approach where our interface design choices ultimately cause us to develop a novel matching algorithm that tolerates the absence of time and amplitude scales on a sketch.

Consider an economist looking for transient historical periods of recession marked by a sharp decrease in gross domestic product (GDP) and then a rise, as well as an increase in unemployment and then a fall. With Qetch, the economist simply sketches two shapes, a rounding bottom (∪) and a plateau (∩), to immediately visualize results, from which she can further refine her query. She can control the order of events across the two time series through the relative horizontal positioning of the sketches on the canvas: overlapping positions indicate that the GDP query pattern should co-occur with the unemployment one. She can easily search for more complex GDP recession patterns such a double-dip recessions by annotating her sketch with a regular expression repetition operator. With existing systems, however, the economist has to consider questions which are not straightforward to answer: *how long was the recession? how big was the fall in gross domestic product? how high was unemployment?* Worse, she may have to start from a known recession period and overlay her sketches on it to find other recession periods. She may also have to use different input modes to specify how queries across different data sets relate. In Qetch, the sketching canvas is the primary and only mode of query specification. This narrows the *gulf of execution* [36] as users sketch the query that matches their mental image of the expected result. Moreover, users assess the accu-



*CHI 2018,* April 21–26, 2018, Montreal, QC, Canada

 ISBN 978-1-4503-5620-6/18/04. . . $15.00

DOI: https://doi.org/10.1145/3173574.3173962

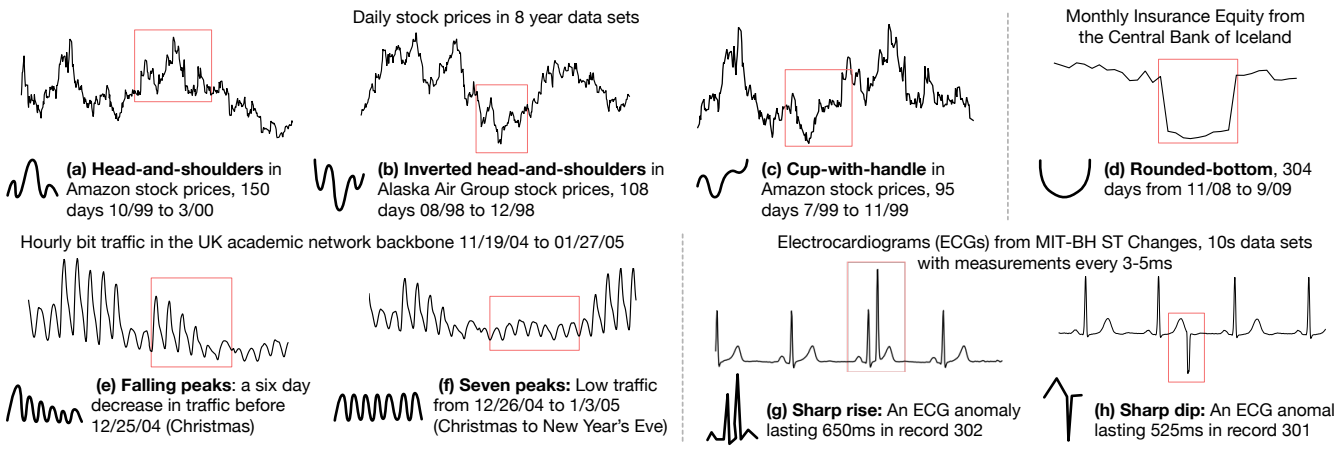



**Figure 1. In a crowd study, we asked 150 crowd workers to sketch the marked regions from different time series. A canonical sketch is displayed for each of the eight visually distinct patterns. Table 1 provides a sample of the crowd workers' sketches.**

racy of a match by visually examining it. Thus, by presenting the sketch query alongside visualizations of results, users can easily understand how to modify their sketches to eliminate unintended matches. This narrows the *gulf of evaluation* [36].

In this paper, we begin by describing an initial crowd-study where we collected a database of human sketches for eight visually distinct patterns for querying different time series (Figure 1). By qualitatively studying these sketches, we identified several sketching behaviors and errors. This study motivates Qetch's interface design and matching algorithm. Given the extensive research in time series querying, we provide a brief description of several querying interfaces and matching algorithms and how they relate to or differ from Qetch. We describe Qetch's query-by-sketch interface, which supports expressive querying features such as regular expressions over sketches and querying across multiple data sets. We describe Qetch's matching algorithm in detail. Finally, we demonstrate the effectiveness of Qetch's interaction features through a user study, we compare the querying performance of Qetch's matching algorithm to commonly-used time series algorithms like dynamic time warping (DTW) and Euclidean distance (ED) on targeted search tasks and we compare the querying performance *as assessed by users* on a variety of exploratory search tasks.

## FROM STUDY TO DESIGN

We conducted a mechanical turk study, where we provided 150 workers with line chart visualizations of eight different time series data from the domains of finance, economics, technology and medicine [37, 5, 14]. Within each chart, we marked a region of interest (see Figure 1 for a description of each time series and region) and asked the workers to sketch the region on an empty canvas with the help of their mouse (or other available input device such as track pad, touch-screen, stylus, etc). The presentation of charts was randomized across workers. The marked-up visualization and the canvas were on separate pages. While workers could navigate back and forth between the marked-up visualization and the canvas, they could not sketch while also viewing the reference region. We chose to set up the experiment as such to ensure that workers were sketching the reference region rather than tracing it. We chose these data sets and reference regions as they represent a diverse set of realistic applications of time series querying[1]. We collected a total of 1200 sketches. We cleaned this data set by asking a different set of crowd workers to confirm if a sketch indeed represents a valid representation of the reference region. We polled a total of 41 workers receiving five distinct votes per sketch. We accepted a sketch as valid if three or more of the five workers voted 'valid'. We discarded 270 sketches from the initial 1200. Table 1 provides a sample of these valid sketches. We then closely examined each valid sketch and across all sketches we observed and coded the following:

***1. A preservation of visually salient perceptual features:*** This observation resonates with findings from visual perception research which concludes that humans decompose complex shapes into parts such as upward or downward slopes, peaks and troughs [30, 20, 43]. We noticed that the more pronounced the feature – the longer the slope, the deeper or wider the trough or peak, the more likely it was sketched (See Table 1). Only 11% of the valid sketches were incomplete: for example, they missed one of the peaks in the seven peaks pattern. Also, 7% of sketches had some noise: tiny bumps on curve segments that are not representative of key perceptual features and may be due to hand jitter.

***2. Non-uniform global scaling:*** Most sketches cannot be uniformly stretched or compressed while preserving aspect ratio to fit the marked region of interest as visualized.

***3. Local distortions:*** Workers exaggerated features such as the width or depth of a peak or trough or the relative difference between the heights of smaller and larger peaks in a pattern. This observation resonates with research on how humans sketch [10].

This study influenced the design of Qetch. We carefully designed Qetch's interface with the sketching canvas as the primary mode of query specification. We lightly smooth sketches and represent them with Bezier curves to eliminate hand jitter and allow easy sketch modification. We provide support for expressing complex queries such as regular expressions over sketches to allow users to easily represent pattern repetitions

[1]The regions marked on the ECGs do not represent arrhythmias: we avoided diagnostic patterns to respect the sensitivities of the workers.

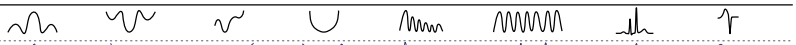
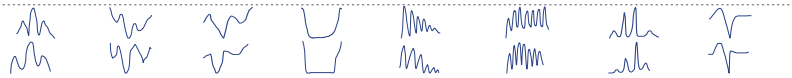
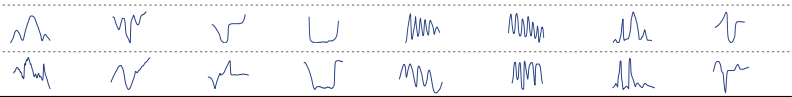
| | |
|---|---|
| Queries |  |
| Sketch Samples |  |
| Typical sketches preserve key perceptual features but have local distortions. |  |
| DTW ranks the reference region at 16+ and Qetch ranks it within 1-15 |  |
| DTW ranks the reference region within 1-15 and Qetch ranks it at 16+ |  |

**Table 1. Samples of sketches drawn by crowd workers for each marked region in Figure 1. Most sketches resemble the first three rows. A few sketches are similar to those from the last row: these *valid* sketches are seemingly poor; they miss or add extraneous features, are incomplete or slightly disagree with the reference region. Note that DTW ranks the reference region in its top 15 results when queried with sketches from the last row.**

or negations, which can be difficult to sketch without such support (recall the missing peaks in the seven peaks pattern). We support ordered querying across multiple time-aligned series. We integrate smoothing during matching and result visualization to allow users to better visualize the similarity between their rough sketch and the matched time series regions. Since sketches are devoid of time scales, we allow users to sort their results by length in addition to distance from sketch.

To support query-by-sketching we require a matching algorithm that is sensitive to the perceptual features of a sketch and gives equal weight to each feature. We, therefore, back Qetch with a matching algorithm that uses curvature [30, 20] to segment the sketch and the data, and matches segments rather than time slices. Our algorithm tolerates sketching errors: it globally rescales the sketch to fit a sequence of time series segments before matching. It also locally rescales each sketch segment to better fit a data segment before computing shape differences. The simple design of Qetch's algorithm allows it to be easily extended to handle regular expressions as well as multiple sketches for querying different time-aligned series.

## RELATED WORKS

Our work draws from time series research on UI design, query specification techniques and query languages, as well as matching algorithms from the database community. We describe these prior works and how they influence or differ from Qetch.

### Time Series Query Specification

Qetch allows users to query time series data with *free-form* sketches on an empty canvas. We broadly classify prior query specification techniques into *sketch-less querying* and *constrained sketching* where users have to (a) overlay sketches over visualizations of time series data, (b) draw shape-restricted query segments, or (c) annotate sketches with amplitude or time scales. As we explain later, such constrained sketching is usually an artifact of the underlying matching algorithm that requires a specification of time or amplitude ranges. Motivated by user-experience, Qetch's interface and matching algorithm supports free-form sketching without constraints.

#### *Sketch-Less Querying*

TimeSearcher introduced *timeboxes* for querying time series data [17, 16, 18, 19]. Timeboxes are rectangular widgets drawn directly on a two-dimensional display of temporal data. TimeSearcher retrieves all time series that pass through one or more timeboxes. Timeboxes are powerful *value-based* querying widgets: for example, one can easily look for stocks whose prices ranged from $1 to $10 in 2016. It is not as easy, however, to specify a *shape-based* query such as a head-and-shoulders pattern with timeboxes.

#### *Constrained Sketching*

**Overlays.** QuerySketch [42] allows users to draw their search pattern on the same display as the data. This design choice of overlaying the query canvas over the data visualization meant that *users had to sketch their patterns with little scaling errors and with pre-defined time and amplitude ranges* — the time and amplitude ranges of the underlying visualized data. This assumption allows the use of standard matching algorithms such as Euclidean distance to find time series similar to the query. Other works have since built on QuerySketch's use of overlays, namely QueryLines [41] and most recently RINSE [45]. Holz and Feiner utilize overlays to derive a user's tolerance of spatial deviations from a sketch by measuring point-by-point distances between the sketch and the underlying region [21].

**Shape-restricted Sketches.** Keogh & Smyth propose a probabilistic pattern matching technique that represents the underlying time series data as segments of piecewise straight lines [27]. They state that there is "considerable psychological evidence ... that the human visual system segments smooth curves into piecewise straight lines" [27]. Given the underlying data representation, they limit query sketches to straight line segments as well. By modifying how users sketch, users can never fully express their intended query. They, however, become more tolerant of matches that fit the shape-restricted sketch even if they do not match their intention. Research on human visual perception does suggest that users mentally decompose complex shapes, such as time series data, into visually salient parts such as peaks, troughs as well as upward- and downward- sloping lines [30, 20, 43] and do so using curvature [20, 30]. *Based on this research, Qetch allows users to freely sketch their query without any shape restrictions but uses curve derivatives to decompose the data and sketch into visually salient segments. The matching algorithm weighs the goodness of a match by how well each of the query and data segments match.*

**Annotated Sketches.** Keogh & Smyth also allow users to visually annotate their segmented linear sketches with horizontal and vertical lines that express the degree of elasticity in amplitude or time that the matching algorithm should allow [27]. Holz & Feiner also allow visual annotations in the form of circles on peaks and troughs. The area of the circle expresses the degree of spatial deviation from the sketch the

matching algorithm should allow [21]. Unlike these works, Qetch utilizes time or value annotations to tighten a query rather than to relax it. Like Qetch, TimeSketch [9] allows users to sketch patterns on an empty canvas. Users, however, have to explicitly specify the time scale of the sketch.

### Query Languages and Regular Expressions

Agrawal et al. first propose the use of a shape definition language (SDL) to describe patterns over time series data [1]. SDL expresses shapes with keywords like up, and its steeper form Up, down, stable, etc. and concatenation, repetition and choice operators. SDL is thus a very powerful language but it suffers from a wide gulf of execution as one has to transform their mental image of a shape into a series of keywords and operators. *Patterns* is a commercial tool that also uses a language of pattern-keywords similar to SDL and supports regular expressions [33]. *Qetch works directly with the sketch allowing users to specify repetition and negation regex operators in the form of sketch annotations.*

### Matching Algorithms

Several surveys compare methods for querying and mining time series data and evaluate their performance on different benchmarks [8, 9, 12, 26]. Ding et al. conclude in their survey that there is no distance measure that is systematically better than the 40-year-old *dynamic time warping* (DTW) measure [8]. Moreover, the relatively simple and straightforward distance measure, *Euclidean distance* (ED) is competitive with DTW and other complex approaches, especially as the size of the data set grows [8]. Thus, we focus only on DTW and ED as benchmarking studies reveal that they not only outperform other algorithms when matching one time series to another [8, 26, 12, 2], but they also show higher subjective accuracy in ranking time series similar to a sketch [9]. Given the plethora of matching algorithms, we provide detailed descriptions of different algorithms in a separate appendix[2] as well as dive deeper into DTW and ED. We briefly discuss the variants of DTW and ED that we use for query-by-sketching.

Both ED & DTW require a query length. With a sliding window size equal to the query length and a step size of 1% of the query length, we extract multiple regions from a time series to compare against the sketch. We scale the sketch to be equal in length [44, 13] and height to each region. We then shift the query such that its mean value is aligned with the region's mean value [44].

***Euclidean Distance (ED).*** ED is the sum of squared differences of values between the query and the region at $k$ sampled points. ED is easy to implement and has linear complexity. However, since distances are computed point-wise and the mapping of a query point to a data point is fixed, ED is sensitive to noise and local time misalignments.

***Dynamic Time Warping (DTW).*** DTW overcomes ED's inability to handle local time misalignments (or warps) by locally stretching or compressing a time series to better match it with another. DTW takes as a parameter a warping window size that limits how much the query can be temporally stretched or compressed. In our experiments we set an unlimited warping window size. Some DTW variants minimize the cost of mismatched end-points, we do not implement this feature as we compare the sketch against several slightly-shifted regions of a data set to find the best match. Also, we find that users care about all perceptual features including those at the periphery of a sketch. In our experiments, we find that additional normalization, such as Z-normalization, beyond global rescaling and offset shifting does not improve DTW's precision. After various experiments with different effectiveness settings for DTW [34], we find that DTW with time and amplitude scaling, offset shifting and an unlimited warping window performs best for sketch to time-series matching. These settings may not work well for time-series to time-series matching.

[2]The appendix is part of the auxiliary material published with this work.

## QUERYING TIME SERIES DATA WITH QETCH

Querying time series data with Qetch involves three steps: (1) Loading, (2) Sketching, and (3) Refining. Users typically iterate over steps two and three. Qetch's novel querying features include (1) annotating sketches with regular expressions to construct more complex or periodic queries, and (2) specifying multiple queries across data sets. Consider the following use case scenarios:

*Example 1:* A quantitative analyst in training, Joe, is examining the effectiveness of a head-and-shoulders pattern,  , at predicting reversals in stock price trends. He has a database of historical stock prices for different companies.

*Example 2:* A physician, Liz, is looking for pacemaker patients who suffered from an arrhythmia by searching a database of electrocardiograms for not-normal rhythms.

*Example 3:* An economist, Bob, is looking for instances where unemployment was correlated with a decline in the equity of insurance companies.

**Step 1. Loading.** Joe first loads one or a *group* of stock price time series into Qetch. He needs to specify the name of the value y-axis (e.g. stock price) and the time or date format of the time x-axis. Qetch immediately stores the data into a relational database and starts a background *pre-processing* thread that iteratively *segments* and *smooths* the data.

***Segmentation.*** To segment a time series, Qetch uses changes in curve monotonicity to determine the end of one segment and hence the beginning of a new segment. Segments with height less than a configurable, data-specific threshold are merged with adjacent segments. This allows users to eliminate micro-patterns, such as those caused by precision fluctuations in measurements from ECG leads, from consideration by the matching algorithm.

***Smoothing.*** For most time series, smoothing is necessary to capture the key patterns of the data, while leaving out noise. Qetch uses the exponential moving average[3] to smooth data. At every iteration, the data is progressively smoothed until the number of segments is reduced by a configurable constant factor from the last iteration (the default setting is 0.1). A slider allows users to see the effect of several smoothing iterations

[3]Compared to the simple moving average, which equally weighs the last $m$ observations, the exponential moving average weighs observations closer to a point more than observations further away; this makes it better suited for averaging values arriving consecutively in time and for capturing peaks and troughs.

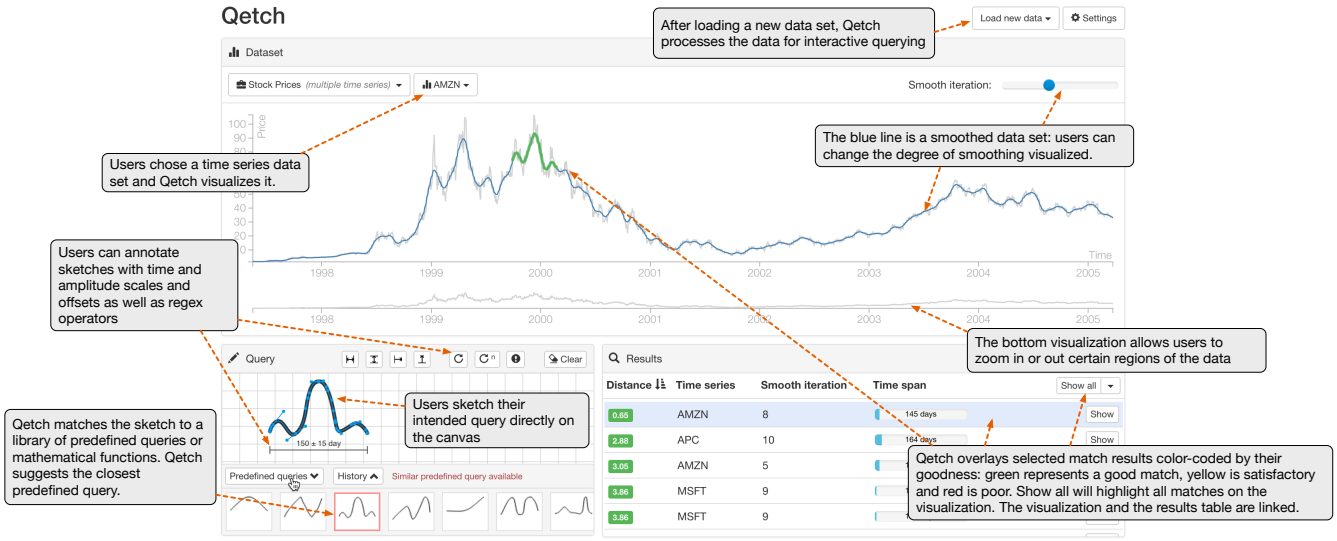



**Figure 2. Qetch's user interface.**

applied to the data. Users can also zoom into different regions of the time series (See Figure 2).

**Step 2. Sketching.** Joe then freely sketches the head-and-shoulders pattern in the empty canvas below the line chart visualization. The canvas has no time or amplitude scales and is independent of the current scales of the visualization. Sketches are slightly smoothed while drawing to remove hand jitter: the sketch is interpolated to create a modifiable Bézier curve. Curve handles can be moved, added or removed. Qetch then executes the matching algorithm and returns an ordered set of all matches for the sketch in the results pane. Matches are color-coded: green for good matches (low distance measures), yellow for fair matches and red for poor matches (high distance measures). Qetch visualizes all the matches over the line chart at the selected smooth iteration. The results table and the line chart visualization are linked such that selecting a match from the table highlights the match on the line chart and selecting a match on the visualization highlights it in the table. Joe can sort results by their distance, time span and degree of smoothing.

**Step 3. Refining.** Joe can further refine his query by annotating it with *query length*, *amplitude range*, *time offset*, or *value offset*. Qetch also maintains a library of mathematical functions (e.g. exponential growth curve $y = a^t$), well-known chart patterns and saved sketches. In addition to matching a user's sketch to the data set, Qetch also matches the sketch to this library and notifies the user of matches found. Joe can preview and select as well as modify patterns from the library to replace his sketch with.

### Regular Expressions over Sketches

Liz, the physician, can use Qetch to find arrhythmias in a data set of patient ECGs. In Figure 3, Liz loads the ECG of a pacemaker patient, she sketches a normal heart rhythm and applies the not operator over the sketch. Qetch finds a match within the ECG, which indicates a premature ventricular contraction. Currently, Qetch only supports three regular expression operators: repeat, repeat exactly $n$ times and not. Users can apply

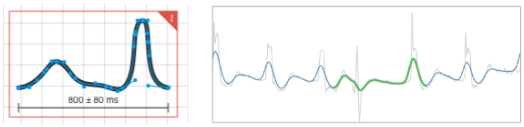



**Figure 3. Regular Expressions with Qetch: Liz searches for abnormal rhythms in a patient's ECG with help of a not operator. Qetch's top result coincides with a premature ventricular contraction.**


**Figure 4. Querying multiple time series with Qetch: Bob simultaneously queries time-aligned insurance equity, and unemployment data sets to find periods where a rise in unemployment coincided with a fall in equity. The relative positioning of the sketches determines whether the matches should overlap or occur in a certain order.**

repeat operators to one or or more *paired segments* of a sketch — a pair of consecutive segments captures a trough or a peak. Users can only apply the not operator once to the entire sketch. While users can nest repeat operators within a not operator and within other repeat operators, they cannot nest a not operator within any operator. By applying regular expression operators directly on the sketch, we preserve the user's mental model of the sketch as the primary query specification mode in Qetch.

### Relative Positioning for Multiple Time Series Queries

Bob, the economist, can search for patterns across different *time-aligned* data sets. In Figure 4, Bob, finds a correlation

between a drop in the equity of insurance companies and an increase in unemployment. Bob controls the order of patterns by the relative positioning of sketches along the time axis. In Figure 4, the sketches overlap on the time axis, thus Qetch will return matches that also overlap in time. Users can specify queries over more than two data sets. We demonstrate through a user study that relative positioning is more effective than directly specifying simplified order constraints of the form query C before query A, query B after query C, etc.

## THE QETCH MATCHING ALGORITHM

*Selecting a match candidate.*

Recall that Qetch segments and smooths time series data. A single time series is represented by multiple time series of increasing degrees of smoothing. Qetch finds matches in each smoothed representation. Each data segment is a curve with either positive or negative derivatives. Similarly, a sketch is segmented into $k$ curves of positive or negative derivatives. We consider a segment whose height is less than 1% of the overall sketch height to be noise, and not a key perceptual feature. Hence, Qetch merges these segments with an adjacent segment.

Without query length, we rely on the number of query segments to determine the number of data segments we consider for a match. Qetch's algorithm slides down the data, one segment at a time, selecting $k$ segments to match to the sketch. If the data has $\mathcal{S}$ smoothed representations and at most $\mathcal{D}$ segments, then the number of matching operations is $O(\mathcal{S} \cdot \mathcal{D})$.

Unlike Qetch, other matching algorithms like ED or DTW have to use a sliding window equal to a user-specified query length to select match candidates. Configuring the step size of the sliding window can be tricky. A small step size can be detrimental to performance as many matches are computed for slightly shifted windows of similar, overlapping time series. On the other hand, a large step size can skip over entire or partial potential matches leading to poor results.

*Global non-uniform scaling.*

A query in Qetch is a scale-less sketch. Consider a sketch of an upward sloping straight line at a 45°angle. One can visualize any upward sloping line segment of a time series to appear to rise at a 45°angle by adjusting the time-scale and amplitude scale accordingly. With such a visualization, the time series is a perfect match for the sketch. This is a key intuition behind Qetch's matching algorithm: the closest match to a sketch devoid of scale is the one that fits the most after we rescale the sketch to fit the match's scales. We compute the following global rescaling factors $(G_x, G_y)$ to rescale the sketch, $Q$, to a match candidate $C$ (Figure 5(a)):

$$G_x = \text{width}(C) \,/\, \text{width}(Q) \qquad G_y = \text{height}(C) \,/\, \text{height}(Q)$$

*Computing the distance measure.*

If humans were precise sketch artists then a standard distance measure such as the Euclidean or Manhattan distance would suffice to measure how close a sketch is to a query. Humans, however, are prone to exaggerate features. For example, a sketch of a head-and-shoulders pattern may exaggerate the relative height difference between the head and the shoulders.

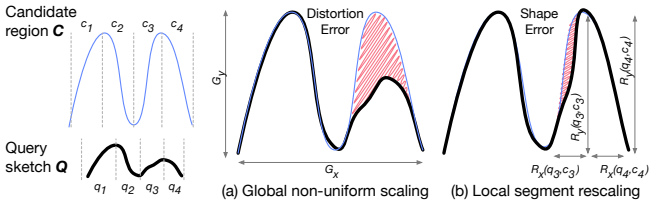



**Figure 5. Global non-uniform scaling and local segment rescaling**

These *local distortion errors* need to be accounted for before measuring the distance between a data segment and a query segment.

**Local Distortion Errors.** Given a match candidate $C$ with $k$ segments $\{c_1, ..., c_k\}$ and a query $Q$ with $k$ segments $\{q_1, ..., q_k\}$. A local distortion error (LDE) is a measure of how much additional rescaling is required to fit a sketch segment $q_i$ to a data segment $c_i$ (Figure 5).

$$R_x(q_i, c_i) = \frac{\text{width}(c_i)}{G_x \ * \ \text{width}(q_i)} \qquad R_y(q_i, c_i) = \frac{\text{height}(c_i)}{G_y \ * \ \text{height}(q_i)}$$

We obtain the local distortion error (or *distortion distance*) from the above rescaling factors $(R_x, R_y)$ as follows:

$$\text{LDE}(q_i, c_i) = \log(R_x(q_i, c_i))^2 + \log(R_y(q_i, c_i))^2$$

A sketch segment that is a factor of four wider than a data segment is just as bad as a sketch segment that is a quarter of the width of a data segment. Thus, we log-normalize the factors and square the result to obtain an appropriate positive error-distance measure. If no local rescaling is required, then the local distortion error is zero as expected.

**Shape Errors.** The difference in shape (or *shape error* SE) between a data and query segment is the Manhattan distance between the two segments, after locally distorting the query segment (Figure 5(b)). Other distance measures (e.g. ED) can also be used here. The query and data are sampled at a fixed time interval (e.g. every 1ms) to produce $N_i$ data points.

$$\text{SE}(q_i, c_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} \left| \frac{G_y * R_y(q_i, c_i) * q_i[j].y - c_i[j].y}{\text{height}(C)} \right|$$

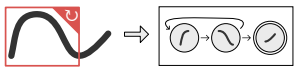
Note that data segments can have different lengths. Since longer data segments have more data points, and thus more differences, we normalize the shape error of each segment by dividing it with the number of data points in it to ensure that longer segments do not overwhelm the overall distance measure. Without such a normalization, smaller micro-patterns may be preferred to larger patterns simply because they have less data points which contribute to fewer differences in the distance measurement. Also, since data segments with larger amplitude variations contribute to larger absolute differences, we normalize each difference by the height of the candidate match. Without such a normalization, Qetch will again prefer patterns of smaller amplitude variations. Thus, the Qetch distance between a query $Q$ and a match candidate $C$ is

$$\text{Dist}(Q, C) = \sum_{i=1}^{k} \text{LDE}(q_i, c_i) + \text{SE}(q_i, c_i)$$

We plan to study how to use multiplicative factors to add more weight to distortion or shape errors. Qetch returns a ranked list of matches from all smoothed series ordered by their distance from the sketch. Matches that fall above a configured distance threshold are eliminated. Only the closest match from matches that overlap across multiple smoothed series is returned. A complexity analysis of Qetch can be found in the appendix within the auxiliary material.

### Evaluating Regular Expressions

Curve segmentation not only enables the efficient selection and matching of candidate matches to user sketches, it also enables the formulation of expressive time series queries with regular expressions. Qetch evaluates regular expressions with the help of a finite-state machine. Each segment in the sketch represents a state in the machine. A sequence of consecutive segments in the sketch is represented by a sequence of connected states. A repeat operator simply adds more transitions to the machine.



Qetch begins at the start state and finds all *good* partial matches for the starting segment (i.e. matches with a distance measure below a certain threshold). At the next state, Qetch filters the partial matches from the previous state keeping only those that can be extended to include the current state's segment. Thus, all possible transitions from a state are explored as long as the matches can be extended. If no good matches are found from extending the partial matches, no further exploration occurs. At the terminal state, all good matches found so far are returned as results. When no partial matches are circulating through the machine, the search ends. Currently, the not operator can only be applied to the entire sketch. After finding all matches $M$ to the sketch that have a distance score below a given pre-configured threshold, the matches $M$ are eliminated from the time series and the remaining contiguous segments are returned as results. Results with larger distances from the sketch are ranked higher.

## EVALUATION

### A User Study of Qetch's Interaction Features

To evaluate Qetch's user interface, we conducted a within-subjects comparative user study of Qetch's novel time series querying features: (i) regular expressions for querying repeated patterns and for anomaly detection versus no regular expressions and (ii) relative positioning of sketches for querying across multiple data sets verses specifying order constraints over sketches. Our goal was to observe query completion times on assigned querying tasks to objectively determine whether Qetch's features improved querying. We also elicited user preferences for the degree of smoothing for which queries should be evaluated and results presented.

#### *Participants & Methods*

We recruited 20 university students and researchers (10 male, 10 female; 17 students, 3 researchers) to evaluate the effectiveness of Qetch's time series querying features. A pre-study survey showed that our subjects had a range of familiarity and expertise with time series data as indicated on a 5-point Likert scale of self-reported familiarity ($\mu = 2.8$, ▮▮▮.). Six of our subjects have also used tools to query or explore time series data (e.g. Google Analytics, matplotlib, Google Charts, Python, STATA, etc.). Our participants were students of, or researchers within different majors including Neuroscience, Economics, Biology, Physics, and Computer Science. None of our subjects had used Qetch prior to the study.

We first presented a 10-minute scripted Qetch tutorial. The tutorial described what are time series data and line chart visualizations and how to (i) load data sets, (ii) query with sketches, (iii) interpret results, and (iv) use advanced features such as regular expressions and querying across multiple data sets. We then allowed the participants to play with the tool for 10 minutes. The users mostly directed this play-time, but we asked users to attempt at least two sample query tasks. In this play time we answered any questions they had about the tool.

We then asked the subjects to attempt a series of querying tasks within 600 seconds (described below) on *synthesized* data sets with and without a particular Qetch feature. Each task involved sketching a given pattern in Qetch with the help of a mouse with a specific feature turned on or off. We opted for mouse input rather than pen-based input to ensure our results generalize; many users may not have used pen-based input devices nor have easy access to them. We chose to synthesize data sets for each task in the comparative user-study to ensure users can easily verify whether they successfully completed a task or not and to ensure that tasks performed with or without a Qetch feature at a certain difficulty level are equivalent. The synthesized data sets also eliminated a user's familiarity with a particular domain from being a confounding variable in our study. Our choice of whether the user attempted the tasks with Qetch's feature or not was randomized as well as our data set choices. This counterbalancing minimized learning effects.

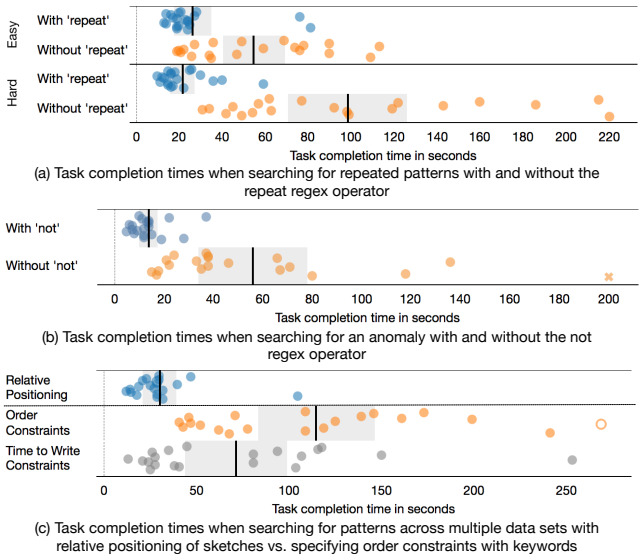
***Task set A: Search for repeated patterns.*** The users looked for a pattern ⋀⋀ that repeated 4 times, or its vertically flipped pattern, ⋁⋁, on a vertically flipped data set, with or without the use of the repeat regex operator. In a *harder* variant, users looked for six repetitions of ⋀⋀ or ⋁⋁. To describe the tasks to the users, we provided printed query sketches.

***Task set B: Detect an anomaly.*** The users looked for an anomalous region, (e.g. ⋀, ⋂) within a data set of recurring peaks (⋂), with or without the use of the not operator. Users did not know the exact shape nor the position of the anomaly, which varied from one task to the other.

***Task set C: Query across multiple data sets.*** The users looked for three patterns across three data sets. We chose a non-linear ordering of the patterns to ensure equal difficulty for specifying the query by relative positioning or ordering constraints. To describe the task to the users, we provided a printout of the query result as visualized by Qetch rather than a sketch of the query so as not to favorably bias query specification with relative positioning as users might copy the order in the sketches directly.

#### *Comparative User Study Results*

Figure 6 summarizes the results of the user study for all task sets. Notice that using Qetch's features, regular expression




(a) Task completion times when searching for repeated patterns with and without the repeat regex operator

(b) Task completion times when searching for an anomaly with and without the not regex operator

(c) Task completion times when searching for patterns across multiple data sets with relative positioning of sketches vs. specifying order constraints with keywords

**Figure 6. Black bars indicate mean task completion times. The shaded regions represent 95% CIs. Dots, representing each user's task completion time, are vertically jittered to minimize occlusion. The cross represents a user who could not complete that task within 10 minutes. Hollow dots indicate tasks completed incorrectly.**

operators and relative positioning of sketches for querying multiple data sets, leads to faster mean task completion times regardless of task difficulty.

To evaluate the effectiveness of the repeat operator, we performed a two-way repeated measures ANOVA with task difficulty and operator-use as independent factors. We log-transformed the completion times to better approximate a normal distribution[4]. We found a significant interaction effect of operator-use and task difficulty, $F_{1,19} = 11.3, p = .003$. We found the following simple main effects: *for both easy and hard tasks, there was a significant difference in completion times between using and not using the* repeat *operator*: $F_{1,19} = 20.5, p < .0005, F_{1,19} = 63.7, p < .0005$. Without the repeat operator, there was a significant difference in task completion times between easy and hard tasks $F_{1,19} = 13.6, p = .002$. When using the repeat operator however, there was no significant difference in completion times between the easy and hard tasks as expected: in both cases the user sketches a relatively simple pattern once and sets the number of repetitions in the repeat operator.

To evaluate the not operator, we performed a one-way repeated measures ANOVA with operator-use as an independent factor. We log-transformed the completion times. *We observe a statistically significant effect of using the* not *operator on task completion times,* $F_{1,18} = 32.5, p < .0005$ [5]. Without access to the not operator, users resorted to different tactics to find the anomaly including (a) visually scanning the entire time series at different zoom levels to manually find the anomaly, or (b) sketching the recurring pattern (a peak) and sorting the results by distance from sketch in descending order.

[4]Since only two levels per factor exist, we assume perfect spherecity.
[5]We dropped a user's completion times as s/he timed out when trying to find the anomaly without the not operator.

Finally, to evaluate the effectiveness of relative positioning of sketches as a technique for querying multiple data sets versus the explicit specification of order constraints with keywords on sketches, we performed a one-way repeated measures ANOVA with technique used as an independent factor. We also log-transformed the completion times and we assume perfect spherecity. *We found a significant effect of technique used,* $F_{1,19} = 68.6, p < .0005$. Figure 6(c) illustrates that the difference in performance is largely explained by the time required to write ordering constraints of the form {Q3 before Q1, Q1 before Q2, ...}: the mean time to write constraints after providing the query sketches for each data set was 71 seconds (the total mean time to complete the task with ordering constraints was 115 seconds), the mean time to query with relative positioning, however, was only 30 seconds ($\approx 115 - 71$ seconds). In a post-study questionnaire, users were asked which method did they prefer for specifying queries over multiple data sets. 18 users of the 20 users preferred relative positioning over the specification of order constraints. Some users elaborated on their preferences: '*constraints are not intuitive, hard to understand*', '*It [would be] hard to write constraints for more than 4 queries*', '*Drawing is much easier.*', '*[specification with order constraints] is a bit more complex, but not complicated*'.

### *Smoothing Preferences*

For each of the eight queries described in Figure 1, we presented the 20 participants with a query sketch, and the data set with the result highlighted in a different color. For fine-grained preference elicitation and comparisons across data sets, we increased the number of smoothing attempts per iteration to ensure that each data set had roughly twenty smoothed series with the exception of the insurance equity data set, which did not have enough data points to allow more than five smoothing iterations. We then asked users to slide three sliders to indicate the minimum, preferred and maximum degree of smoothing of the data set and the highlighted result. Figure 7 summarizes the results. We observe that for most queries, users want a minimum non-zero degree of smoothing to exist. We also observe that users set a maximum degree of smoothing well below 50% of the highest possible degree of smoothing supported for a particular data set for most queries. The preferred degree of smoothing is somewhere in between the minimum and the maximum. *We find that Qetch's smoothing choices are within a 95% confidence interval of the mean preferred smoothing degree for six of the eight queries.* This affirms that presenting users with results smoothed at the level at which they best match the query sketch is an effective presentation choice. Note that for the rounded-bottom query ∪ on the insurance equity data set, the highest degree of smoothing supported is only five iterations and 65% of the users chose a smoothing level of one as their preferred smoothing level and 35% chose no smoothing as Qetch chose. For the sharp dip query ⋀, we contend that Qetch still chooses a degree of smoothing that is roughly within one standard deviation of the preferred mean smoothing degree selected by users.

### *Qualitative Evaluation of the* Qetch *Interface*

We also conducted a post-study questionnaire to gain some qualitative feedback on Qetch. Overall, users found Qetch

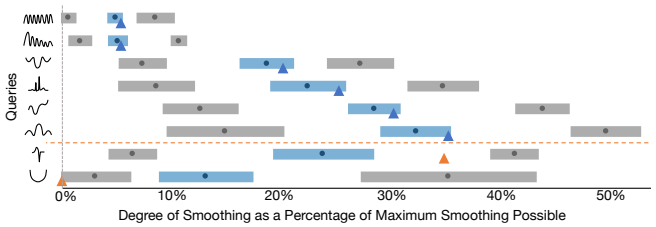



**Figure 7. For each query, we present 95% CI bars for the minimum, preferred, and maximum mean smoothing degrees as selected by users. The triangle glyph marks Qetch's degree of smoothing.**

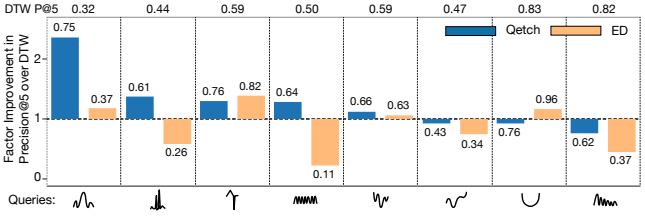



**Figure 8. Bars mark factor improvements in precision@5 of Qetch, and ED over DTW. Bars at the reference line of 1 indicate no improvement in precision. We see that Qetch provides at least a factor of two improvement in precision@5 over DTW on the head-and-shoulders query. For reference, each bar is additionally labeled with the precision@5 value of the different algorithms and DTW's precision@5 value is listed at the top of the figure.**

easy to use ($\mu = 4.1$, ). They found query-by-sketching an intuitive and effective paradigm for querying time series data ($\mu = 4.2$, ), with only two users wishing for more formulaic methods for specifying queries. Most users found Qetch interactive and responsive enough to freely correct sketching errors ($\mu = 4$, ). One user commented that the '*NOT was a little bit slow, it freezed [sic] for one second*'. In general, users found the table visualization and the ordering of results by different attributes effective ($\mu = 4.6$, ; $\mu = 4.7$, ). Users responded to a general feedback question on Qetch. One user stated "*I wish I was able to zoom into my drawings and correct the mistakes rather than start over.*" Three other users expressed a similar interest in editing shape vectors or drawing sketches with multiple connected segments instead of a '*single click and hold*' interaction. Four users expressed a desire for a better input device such as a '*good stylus*', '*sketchpad*', '*touchscreen*' or a '*better mouse*'. One user stated that the not operator was only effective '*when the pattern is extremely clear, otherwise is not useful at all*', we assume the user was referring to the anomalous pattern being clearly distinguishable from the underlying data set. While two users found Qetch sensitive — '*[Qetch ] gets confused very easily if the drawing is not accurate enough*', '*the tool is powerful enough to find the anomalies, but the sketch has to be precise*' — two other users found Qetch forgiving: '*I realized that the program is forgiving if it [the sketch] is not exact, which makes it easier*', '*I like how it automatically completed & identified shapes and also ignored mistakes that were obvious*'.

We observed during our experiments, that users occasionally change the data visualization scale in order to better assess Qetch's proposed matches to their scale-less sketch. We plan to narrow this gulf of evaluation in the future by automatically rescaling visualized results to better match the sketch. Note that the time spent comparing matches at a scale different from the sketch is often negligible relative to the time that would have been spent specifying queries of multiple sizes especially if the user has no set query size in mind.

### Qetch's Performance on Search Tasks

We evaluate Qetch's performance on two types of search tasks: (1) *targeted search*, where users search the data set for a specific region, and (2) *exploratory search* where users search for several regions in the data set that match their sketch.

#### *Targeted Search*

We compare the precision of Qetch to both DTW and ED. In this experiment, the query length determined the size of the sliding window for both DTW and ED. The step size was set to 1% of the query length. Further experimental setup details such the number of data points and segments per data set, the number of sketches per query, and other data and query statistics can be found in the appendix.

In our initial crowd study, crowd workers provided sketches that best represented a marked reference region in eight different data sets (See Figure 1). We evaluated how often each algorithm placed this reference region in its top 5 results when the sketch was used to query the time series. Let $S_i$ be the set of sketches for one of the eight reference regions ($i = 1, ..., 8$). Let $\mathbf{rank}_i(s)$ be the rank of the reference region when a sketch $s$ is used to query the reference region's data set. Precision@5($S_i$) is defined as:

$$\frac{1}{|S_i|} \sum_{s \in S_i} \begin{cases} 1, \mathbf{rank}_i(s) \leq 5 \\ 0, \text{otherwise} \end{cases}$$

Precision@5 ranges from 0 to 1, with higher precision values indicating that more sketches caused the target reference region to appear in the top 5 results for a given matching algorithm.

In Figure 8, we find that Qetch outperforms DTW and ED with a few exceptions. For the rounded-bottom ( ) pattern, there are many regions in the data that can match a simple trough pattern, and without query length, Qetch cannot filter out these matches as well as the other algorithms. A similar problem occurs with the cup-with-handle ( ) query. Note that the falling-peaks ( ) pattern queries the network traffic data set which is mostly a periodic data set of five weekdays of high traffic peaks and two weekends of low traffic peaks: Qetch often matches the falling-peaks sketch to four high weekday peaks followed by two low weekend peaks due to the relatively small distortion errors caused by the three middle peaks. We also computed the mean reciprocal rank [6] and the factor improvement of Qetch over DTW and ED are equivalent to those for precision@5. Hence, we do not include those results for brevity.

#### *Exploratory Search*

We recruited 16 university students, faculty, and staff (8 male, 8 female; 4 students, 9 researchers, 3 staff). Their background was of a similar make-up to the previous study. None of our subjects had used Qetch prior to the study. We administered the same training protocol as before. We asked users to perform the following two query tasks: (1) A  pattern lasting

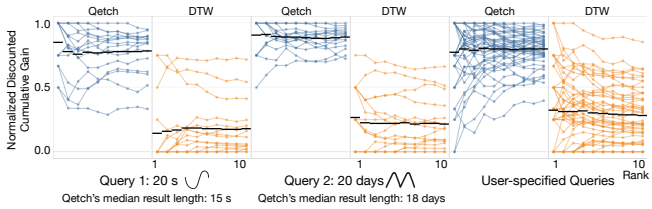



**Figure 9. Each line represents the normalized discounted cumulative gain (NDCG) for each user query for Qetch and DTW. The black bars mark the average NDCG. For queries 1 and 2, we provide the median result length of Qetch. Details of the user-defined queries and their Qetch results can be found in the appendix.**

roughly 20 seconds on the MIT-BIH heart rate series 1 [32], and (2) A ⋀⋀ pattern lasting roughly 20 days on the mean daily temperature in Saugeen river [15]. We then asked users to freely sketch three queries of their choice on the following data sets: Alaska Air Group's (ALK) stock prices, (2) Anadarko Petroleum Corporation's (APC) stock prices and (3) the MIT-BIH heart rate series 3 [32]. Users were allowed as many attempts as they wished before they finalized their three free queries. Further details of the data sets and the range of user-specified queries can be found in the appendix.

For each user query, we presented the top-10 results of DTW and Qetch. To minimize user fatigue, we did not include ED in our comparison. The presentation of the top-10 results for each algorithm was randomized across users. For Qetch, we presented the smoothed overlay in addition to the underlying data region for each query result. Note that DTW does not utilize smoothing and hence cannot benefit from this Qetch feature.

The users rated the relevance of each result on a 5-point scale ranging from bad (1) to perfect (5). We evaluated the effectiveness of DTW and Qetch with the popular *normalized discounted cumulative gain* (NDCG) measure [7, 22]. The discounted cumulative gain (DCG) is a measure of the usefulness, or gain, from examining a region in the time series. The gain of each ranked result is its user-rating weighted or discounted by its rank in the overall result and the DCG at rank $i$ is the aggregate of all discounted gains of each result up to result $i$. The NDCG normalizes the DCG at each rank with the *ideal* DCG value assuming a perfect ranking. We determine the ideal DCG by aggregating the results of both DTW and Qetch for each user query and ordering the results from highest rated to lowest rated. NDCG ranges from 0 indicating a poor ranking relative to the ideal ranking of results to 1 indicating a perfect ranking. Figure 9 provides the NDCG for each user query and the average NDCG for each of the query types and algorithms. Qetch noticeably outperforms DTW across all queries.

Users attributed Qetch's superior performance to two aspects. First, Qetch's smoothing choices and its presentation of smoothed results. One may assume that smoothing is independent of the matching algorithm and the querying interface; after all, one can apply smoothing post hoc to a region identified as a match by DTW. This, however, is not trivial as it is difficult to determine the right degree of smoothing and smoothing might in fact lead to a worse DTW distance measure. The careful integration of smoothing into Qetch's matching algorithm and interface leads to its overall effectiveness. Second, Qetch gives equal importance to each perceptual feature within a sketch. As two users expressed: '*Tool A [Qetch ] was better at finding smaller features of the query*', '*When tool B [DTW ] is used, it tends to look at the bigger picture and ignore minor peaks [in the sketch]*'.

## LIMITATIONS AND FUTURE WORK

Rakthanmanon et al. state that "after an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount on reproducible experiments" [40]. We agree with this statement; Qetch does not outperform DTW in standard time-series benchmarking tests. Qetch's distance measure is a poor choice for measuring the distance of one time-series region to another. Thus, it is also a poor choice for other time-series applications such as motif-discovery (finding recurring patterns in a time series). It is, however, suitable for the specific use-case scenario of matching a rough, hand-drawn, scale-less sketch to a time series. DTW, however, and its well-studied variants were not designed for this use-case. We believe that by re-examining query-by-sketching, we open up venues for further research into matching algorithms.

For the small to medium scale time series we studied, Qetch provides interactive performance with high precision. A common finding on large-scale data sets (millions to trillions of data points), is that differences in accuracy/precision between time-series matching algorithms diminish [8]. With larger data sets, the opportunity to find good matches with all three algorithms increases, which leads to good precision@$k$ in spite of the matching algorithm used. We argue that for many users who work with relatively small data sets, slightly worse precision can be detrimental to the overall user-experience and we should strive to build interfaces that work well for data sets of all sizes. We plan to explore techniques to improve Qetch's scalability by exploiting the inherently embarrassingly parallel nature of searching across the multiple smoothed and partitioned series and by rapidly pruning regions from consideration by indexing the gradients of different segments.

## CONCLUSION

In this paper, we introduced Qetch, a query-by-sketch tool for time series data. We conducted a crowd study to learn how humans sketch time series. We observed that participants often preserve and exaggerate the visually salient features of the reference time series they are sketching. We designed Qetch's matching algorithm to consider and tolerate such distortions. As our evaluation demonstrates, Qetch's sketch-centric design is powerful and expressive: through sketch annotations, users can effectively construct complex regular-expression queries and queries over multiple time-aligned series. Qetch outperforms standard algorithms on targeted and exploratory search tasks. Finally, we publicly release our crowd-sourced data set of sketches and source code [35].

**Acknowledgments:** This work was partially supported by NSF IIS-1420941.

## REFERENCES


1. Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zaït. 1995. Querying Shapes of Histories. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 502–514.
2. Gustavo E.A.P.A. Batista, Xiaoyue Wang, and Eamonn J. Keogh. 2011. A Complexity-Invariant Distance Measure for Time Series. In *Proceedings of the 2011 SIAM International Conference on Data Mining*. 699–710. DOI: http://dx.doi.org/10.1137/1.9781611972818.60
3. Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (AAAIWS'94)*. AAAI Press, 359–370.
4. Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. H. Tung. 2007. SpADe: On Shape-based Pattern Detection in Streaming Time Series. In *2007 IEEE 23rd International Conference on Data Engineering*. 786–795. DOI:http://dx.doi.org/10.1109/ICDE.2007.367924
5. P. Cortez, M. Rio, M. Rocha, and P. Sousa. 2006. Internet Traffic Forecasting using Neural Networks. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. 2635–2642. DOI: http://dx.doi.org/10.1109/IJCNN.2006.247142
6. Nick Craswell. 2009. *Mean Reciprocal Rank*. Springer US, Boston, MA, 1703–1703. DOI: http://dx.doi.org/10.1007/978-0-387-39940-9_488
7. W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 283. Addison-Wesley Reading.
8. Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proc. VLDB Endow.* 1, 2 (Aug. 2008), 1542–1552. DOI: http://dx.doi.org/10.14778/1454159.1454226
9. Philipp Eichmann and Emanuel Zgraggen. 2015. Evaluating Subjective Accuracy in Time Series Pattern-Matching Using Human-Annotated Rankings. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 28–37. DOI: http://dx.doi.org/10.1145/2678025.2701379
10. Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph.* 31, 4, Article 44 (July 2012), 10 pages. DOI: http://dx.doi.org/10.1145/2185520.2185540
11. Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2009. PhotoSketch: A Sketch Based Image Query and Compositing System. In *SIGGRAPH 2009: Talks (SIGGRAPH '09)*. ACM, New York, NY, USA, Article 60, 1 pages. DOI: http://dx.doi.org/10.1145/1597990.1598050
12. Philippe Esling and Carlos Agon. 2012. Time-series Data Mining. *ACM Comput. Surv.* 45, 1, Article 12 (Dec. 2012), 34 pages. DOI: http://dx.doi.org/10.1145/2379776.2379788
13. Ada Wai-Chee Fu, Eamonn Keogh, Leo Yung Lau, Chotirat Ann Ratanamahatana, and Raymond Chi-Wing Wong. 2008. Scaling and Time Warping in Time Series Querying. *The VLDB Journal* 17, 4 (July 2008), 899–921. DOI:http://dx.doi.org/10.1007/s00778-006-0040-z
14. Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. Physiobank, Physiotoolkit, and Physionet Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101, 23 (2000), e215–e220.
15. Keith W Hipel and A Ian McLeod. 1994. *Time series modelling of water resources and environmental systems*. Vol. 45. Elsevier.
16. Harry Hochheiser. 2002. Interactive Querying of Time Series Data. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems (CHI EA '02)*. ACM, New York, NY, USA, 552–553. DOI: http://dx.doi.org/10.1145/506443.506477
17. Harry Hochheiser and Ben Shneiderman. 2001. Interactive Exploration of Time Series Data. In *Proceedings of the 4th International Conference on Discovery Science (DS '01)*. Springer-Verlag, London, UK, UK, 441–446.
18. Harry Hochheiser and Ben Shneiderman. 2002. Visual queries for finding patterns in time series data. *University of Maryland, Computer Science Dept. Tech Report, CS-TR-4365* (2002).
19. Harry Hochheiser and Ben Shneiderman. 2004. Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration. *Information Visualization* 3, 1 (March 2004), 1–18. DOI: http://dx.doi.org/10.1145/993176.993177
20. Donald D Hoffman and Manish Singh. 1997. Salience of visual parts. *Cognition* 63, 1 (1997), 29 – 78. DOI: http://dx.doi.org/10.1016/S0010-0277(96)00791-3
21. Christian Holz and Steven Feiner. 2009. Relaxed Selection Techniques for Querying Time-series Graphs. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 213–222. DOI: http://dx.doi.org/10.1145/1622176.1622217
22. Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. DOI: http://dx.doi.org/10.1145/582415.582418
23. Eamonn Keogh. 2003. *Efficiently Finding Arbitrarily Scaled Patterns in Massive Time Series Databases*. Springer Berlin Heidelberg, Berlin, Heidelberg, 253–265. DOI:http://dx.doi.org/10.1007/978-3-540-39804-2_24

24. Eamonn Keogh. 2008. *Indexing and Mining Time Series Data*. Springer US, Boston, MA, 493–497. DOI: http://dx.doi.org/10.1007/978-0-387-35973-1_598

25. Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2004. Segmenting time series: A survey and novel approach. *Data mining in time series databases* 57 (2004), 1–22.

26. Eamonn Keogh and Shruti Kasetty. 2002. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. ACM, New York, NY, USA, 102–111. DOI: http://dx.doi.org/10.1145/775047.775062

27. Eamonn Keogh and Padhraic Smyth. 1997. A Probabilistic Approach to Fast Pattern Matching in Time Series Databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*. AAAI Press, 24–30.

28. Eamonn Keogh, Li Wei, Xiaopeng Xi, Michail Vlachos, Sang-Hee Lee, and Pavlos Protopapas. 2009. Supporting Exact Indexing of Arbitrarily Rotated Shapes and Periodic Time Series Under Euclidean and Warping Distance Measures. *The VLDB Journal* 18, 3 (June 2009), 611–630. DOI: http://dx.doi.org/10.1007/s00778-008-0111-4

29. Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. 2012. Exploring Collections of 3D Models Using Fuzzy Correspondences. *ACM Trans. Graph.* 31, 4, Article 54 (July 2012), 11 pages. DOI:http://dx.doi.org/10.1145/2185520.2185550

30. Nicholas Kong and Maneesh Agrawala. 2009. Perceptual Interpretation of Ink Annotations on Line Charts. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 233–236. DOI: http://dx.doi.org/10.1145/1622176.1622219

31. Jefrey Lijffijt, Panagiotis Papapetrou, Jaakko Hollmén, and Vassilis Athitsos. 2010. Benchmarking Dynamic Time Warping for Music Retrieval. In *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '10)*. ACM, New York, NY, USA, Article 59, 7 pages. DOI: http://dx.doi.org/10.1145/1839294.1839365

32. George B. Moody and M.D. Ary L. Goldberger. 2017. MIT-BIH Heart Rate Time Series. (July 2017). http://ecg.mit.edu/time-series/

33. Jeffrey P. Morrill. 1998. Distributed Recognition of Patterns in Time Series Data. *Commun. ACM* 41, 5 (May 1998), 45–51. DOI: http://dx.doi.org/10.1145/274946.274955

34. Abdullah Mueen and Eamonn Keogh. 2016. Extracting Optimal Performance from Dynamic Time Warping. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 2129–2130. DOI:http://dx.doi.org/10.1145/2939672.2945383

35. New York University Abu Dhabi, Design Technology Lab. 2018. Qetch source code repository. (January 2018). https://github.com/dtl-nyuad/qetch

36. Donald A. Norman. 2002. *The Design of Everyday Things*. Basic Books, Inc., New York, NY, USA.

37. Central Bank of Iceland. 2016. Insurance companies statistics. (Jun 2016). http://www.cb.is/statistics/statistics/2016/08/15/Insurance-companies/

38. Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. 2011. Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Trans. Graph.* 30, 4, Article 33 (July 2011), 10 pages. DOI: http://dx.doi.org/10.1145/2010324.1964928

39. Sanghyun Park, Sang-Wook Kim, and Wesley W. Chu. 2001. Segment-based Approach for Subsequence Searches in Sequence Databases. In *Proceedings of the 2001 ACM Symposium on Applied Computing (SAC '01)*. ACM, New York, NY, USA, 248–252. DOI: http://dx.doi.org/10.1145/372202.372334

40. Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 262–270. DOI: http://dx.doi.org/10.1145/2339530.2339576

41. Kathy Ryall, Neal Lesh, Tom Lanning, Darren Leigh, Hiroaki Miyashita, and Shigeru Makino. 2005. QueryLines: Approximate Query for Visual Browsing. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 1765–1768. DOI: http://dx.doi.org/10.1145/1056808.1057017

42. Martin Wattenberg. 2001. Sketching a Graph to Query a Time-series Database. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. ACM, New York, NY, USA, 381–382. DOI: http://dx.doi.org/10.1145/634067.634292

43. M Wertheimer. 1938. Laws of organization in perceptual forms (partial translation). *A Sourcebook of Gestalt Psychology* (1938), 71–88.

44. Yunyue Zhu and Dennis Shasha. 2003. Warping Indexes with Envelope Transforms for Query by Humming. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*. ACM, New York, NY, USA, 181–192. DOI: http://dx.doi.org/10.1145/872757.872780

45. Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. 2015. RINSE: Interactive Data Series

Exploration with ADS+. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1912–1915. DOI:
**http://dx.doi.org/10.14778/2824032.2824099**