Semantics and Anomaly Preserving Sampling Strategy for Large-Scale Time Series Data

SHIBBIR AHMED, Iowa State University, USA
MD JOHIRUL ISLAM, Amazon Inc, USA
HRIDESH RAJAN, Iowa State University, USA

We propose *PASS*, a $O(n)$ algorithm for data reduction that is specifically aimed at preserving the semantics of time series data visualization in the form of line chart. Visualization of large trend line data is a challenge and current sampling approaches do produce reduction but result in loss of semantics and anomalous behavior. We have evaluated *PASS* using seven large and well-vetted datasets (Taxi, Temperature, DEBS challenge 2012-2014 dataset, New York Stock Exchange data, and Integrated Surface Data) and found that it has several benefits when compared to existing state-of-the-art time series data reduction techniques. First, it can preserve the semantics of the trend. Second, the visualization quality using the reduced data from *PASS* is very close to the original visualization. Third, the anomalous behavior is preserved and can be well observed from the visualizations created using the reduced data. We have conducted two user surveys collecting 3,000+ users' responses for visual preference as well as perceptual effectiveness and found that the users prefer *PASS* over other techniques for different datasets. We also compare *PASS* using visualization metrics where it outperforms other techniques in five out of the seven datasets.

CCS Concepts: • **Human-centered computing** → *Information visualization;*

Additional Key Words and Phrases: Time series data visualization, sampling, semantics, anomaly

ACM Reference format:
Shibbir Ahmed, Md Johirul Islam, and Hridesh Rajan. 2022. Semantics and Anomaly Preserving Sampling Strategy for Large-Scale Time Series Data. *ACM/IMS Trans. Data Sci.* 2, 4, Article 41 (March 2022), 25 pages.
https://doi.org/10.1145/3511918

1 INTRODUCTION

Visualization of time series data is essential in several domains [29, 35], but challenging [11]. In particular, the size of the data is a significant barrier to interactive visualization [4, 11]. Data reduction is the key strategy to address this problem. Data reduction is widely used in approximate query processing [5], visualization [36], spatial [28] and temporal [9] data analysis. These data reduction strategies work well for map plot, scatter plot, etc., but they are not a good fit for visualization

This material is based upon work supported by the National Science Foundation under Grant CNS-21-20448, CCF-15-18897, CNS-15-13263, CCF-19-34884. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Authors' addresses: S. Ahmed and H. Rajan, Department of Computer Science, Iowa State University, 226 Atanasoff Hall, 2434 Osborn Dr, Ames, IA 50011-1090, USA; emails: {shibbir, hridesh}@iastate.edu; Md J. Islam, Amazon Inc, 11501 Alterra Pkwy, Austin, TX 78758, USA; email: jhislam@amazon.com.

2577-3224/2022/03-ART41 $15.00
https://doi.org/10.1145/3511918

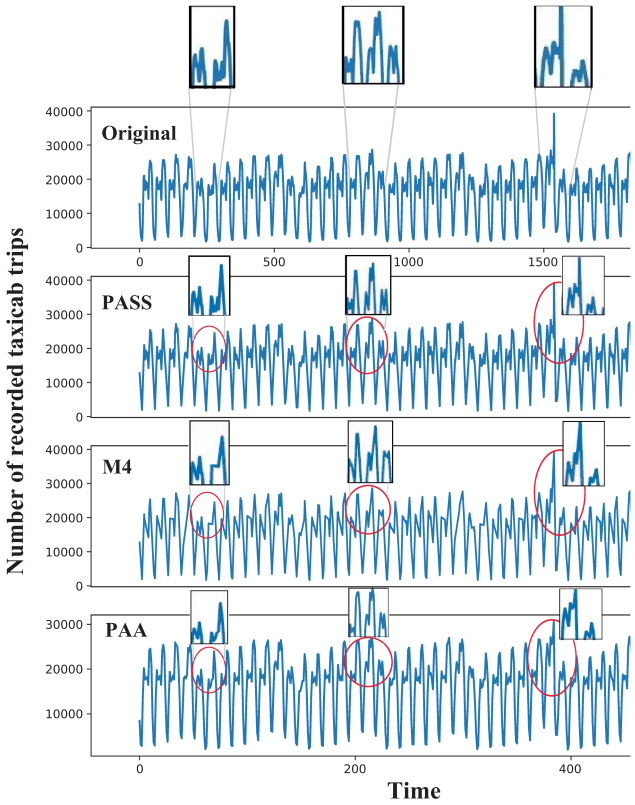


Fig. 1. The time series plots of the volume of taxicab trips in New York City in a two month period in 2014.

of time series data because they can introduce visualization errors and lose trend semantics [29]. (We present a detailed comparison in Section 5.) Due to the loss of time series data visualization semantics along with the loss of ordering and loss of some important anomalies, the goal of exploratory data analysis can be lost. These weaknesses can affect the accuracy and effectiveness of downstream exploratory data analysis tasks that rely on time series data visualization. This is because the observations obtained by reviewing the time series visualization of reduced data might not agree with observations obtained by reviewing the time series visualization of the original data. Some techniques especially for time series data reduction like M4 [29], MinMax [30], and PAA [32] perform very well in producing loss-less visualization but still may lose some semantics that could impact the downstream analysis of trends and anomalies. To illustrate, consider Figure 1 that visualizes the time series plots of Taxi data from [37]. The time series chart in the first row of Figure 1 illustrates the original data where three random locations of the trend are zoomed in. The visualization in the third and the fourth row displays data sampled using M4 and PAA. Readers will note that the trends and anomalies are distorted in the bottom two charts. For example, in the left-most zoomed location, M4 produces a flat horizontal line before the last peak losing the structure of the original chart. The visualization of the data produced by sampling using PAA performs similarly.

These prior works cannot be faulted since their goal is not to preserve trends; however, if exploratory data analyses relied on the presence of these trends, the data analyses and subsequent processes could be misled by their absence. In visualizing large-scale time series data, an ideal reduction strategy should both (a) select fewer number of tuples to reduce the size as well as (b)

preserve the semantics of time series data and anomalous behavior in the visualization. If the semantics produced by sampled data is similar to the original chart, then the analysis or meaning extracted from sampled data exactly match to those found from original data. This goal is very important for domains where anomalous behavior is possible and important to observe, e.g., sudden rise or fall in the stock prices, anomalies in vital sign monitoring of patients, labeling large scale EEG time-series data [8], detecting temporal patterns of viral shedding and transmissibility of COVID-19 [22], and analyzing and forecasting the novel coronavirus COVID-19 cases using robust time series models [49].

1.1 PASS

We present a novel approach, which we call ***PASS* (Preserving Anomaly and Semantics Sampling)**, for reducing time series data where we guarantee that the sampling will preserve the semantics of time series data visualization. *PASS* also reduces the visualization error.

The key insight underlying *PASS* is to split the big time series data into windows such that data within a window are sufficiently close to each other, and then select the beginning and endpoints of the window. More precisely, *PASS* splits the entire range of data into windows such that the deviation angle between two lines, one drawn from the beginning point of a window to the endpoint of the window and the second one drawn from the beginning point of a window to any intermediate point in the window, is less than θ (a threshold tunable by the user). As only the beginning and endpoints of a window are selected in the sample, and intermediate points are discarded, data reduction is achieved. Since all intermediate points with a deviation angle of more than θ are not present within a window, trend line changes are preserved within a window.

To illustrate, we can consider Figure 1 that visualizes Taxi data [37]. The chart in the Figure 1 shows the original data, and other charts created by first applying the *PASS* and then visualizing the reduced data. The trends highlighted in the original plot are all present in the reduced plot in Figure 1. *PASS* achieves data reduction as well as preserves time series data visualization semantics.

1.2 Contributions

The main contributions of this work include:

- A *novel sampling strategy called PASS*. The sampling strategy preserves the anomalies, operates in linear time, and can achieve a very low error rate.
- A *strategy to split the dataset into windows*, based on preselected threshold and use them to create the sample that preserves semantics.
- A *modified gradient descent algorithm to select the best threshold* to minimize the error and to select the best threshold given a sampling rate.
- A notion of quantifying the semantics preservation of time series data computing correlation and quantifying the anomaly preservation with a composite metric using *MSE*, *SSIM*, *DSSIM*, and *PSNR*.
- An experimental evaluation using publicly available datasets [18] to assess the applicability and effectiveness of *PASS* for a variety of time series data.

2 PROBLEM FORMULATION

We now define the problem statement and terminologies.

2.1 Problem Statement

We denote time series dataset as D, set of windows as W, reduced dataset as S, i^{th} window in the set W as W_i, threshold as θ used in the *PASS* algorithm, square error between S and D as E for all tuples in D, and i^{th} point in the dataset as p_i.

Data reduction problem. Given a time series dataset $D = \{(t_1, y_1), (t_2, y_2), \ldots (t_n, y_n)\}$, where i^{th} data point (t_i, y_i) is a tuple comprising of time t_i and the corresponding value of interest y_i at that time, the goal of data reduction is to reduce it to a smaller subset S such that the downstream data analyses applied to a time series visualization of S are: (i) cheaper to apply because the size of S is smaller compared to the size of D, and (ii) produce observations that preserve the semantics as if the analyses were applied to D.

The notion of preserving the semantics and anomaly of an arbitrary data analysis task is challenging. Instead of giving a guarantee over resulting analyses, we focus on the shape of the time series visualization drawn using $S \subset D$. In this context, shape of the time series visualization refers to the structural properties of that visualization. Our intuition is that if the shape of the visualization constructed using D is identical to the visualization constructed using S, then downstream analyses of the visualization that relies on observing patterns and anomalies would produce identical results. The kind of essential semantics and anomaly that our work tends to preserve has been explained in Section 2.2.

2.2 Semantics and Anomaly Preservation

Semantics Preservation. Wang et al. introduced the notion of the semantics of time series data in terms of correlation [62]. The authors modeled the time series chart to understand the system dynamics using pattern-based correlation detection. If a time series chart has similar semantics to another one, those charts are more likely to be derived from the systems with the same dynamics. By time series semantics, the authors indicated similar subsequences present within the line trends where the notion of similarity is quantified by correlation. We also compute correlation to quantify semantics across different windows of two line charts at the same time intervals.

Given two line charts V_1 and V_2, generated through sampling from V_0, V_1 is said to preserve the semantics of V_0 better than V_2 if,

$$corr(V_1, V_0) > corr(V_2, V_0) \tag{1}$$

Here, $corr(V_i, V_0)$ denotes the correlation of line chart V_i and the original line chart V_0.

Anomaly Preservation. Prior works have utilized metrics *MSE* [29], *SSIM* [63], *DSSIM* [29], and *PSNR* [29] to computationally measure the similarity of two images. Our intuition is that the more similar in context of shape of visualization of a line chart V_1 to V_0 is, the more structural properties with anomaly is preserved in V_1 with respect to V_0. So, we introduce a composite metric to measure the anomaly preservation capability of a data reduction technique. As the similarity of two images is directly proportional to *PSNR* and *SSIM* but indirectly proportional to *DSSIM* and *MSE*, a relation can be obtained as shown in Equation (2). All these metrics are formally described in Section 4.1.2.

$$AnomalyPreservation \propto \frac{PSNR \times SSIM}{DSSIM \times MSE} \tag{2}$$

According to the Equations (7), (8), (9), and (10) mentioned in Section 4.1.2, the unit of *MSE* is $pixel^2$, unit of *PSNR* is decibel (dB) and no units for computing *SSIM* and *DSSIM* as those are ratios of similar units. The lowest value of *MSE* and *DSSIM* can be 0, so we take the inverse of the Equation (2) to form Equation (3). For simplicity, we consider proportional linear relationship of the metrics and constant as 1 $pixel^2 db^{-1}$ when all the metrics are normalized to define anomaly preservation score (*APScore*) as follows,

$$APScore = \frac{DSSIM \times MSE}{PSNR \times SSIM} \tag{3}$$

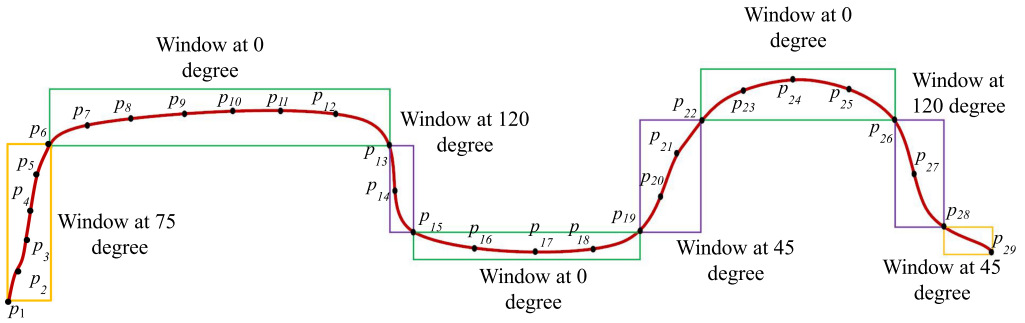


Fig. 2. Splitting a time series visualization into windows.

So, if *APScore* is lower, more anomaly is preserved in the sampled line trend compared to the original line trend and vice versa. We have experimentally shown in Section 4.2.2 that *APScore* is good at measuring the capability of preserving anomalies in the time series data after sampling.

3 PASS METHODOLOGY

PASS (Preserving Anomaly and Semantics Sampling) is a specialized data reduction and sampling strategy that reduces data for the visualization of large-scale time series data as a line chart. Given a dataset D, we first split the dataset into a number of windows having similar properties in terms of trend and angular orientation. Figure 2 represents an example line chart of a dataset.

This dataset can be split into 7–8 windows depending on angular orientation. This strategy ensures that minimum, maximum, and important anomalous behaviors are not lost from trend while reducing the amount of data. After selecting the set of windows W from dataset, we can keep only first and last points from each window. It achieves reduction in the dataset if the window consists of a large number of tuples, which is prevalent in big datasets.

To select windows, we use a tunable parameter *threshold* represented as θ. We assume that the current point belongs to the running window if its deviation from current window orientation is less than θ. Finally, we select a downsample of the original dataset with the help of windows.

Definition 3.1 (PASS Problem). Given a time series dataset D and a threshold θ, the *PASS* problem is to select S such that $S \subset D$ and square error E is minimized to preserve the semantics of time series visualization as well as providing desired reduction in data, i.e., $|S| < |D|$ and the reduction ratio $\frac{|D|}{|S|}$ can be controlled by tuning threshold θ.

Example 3.1 (Example:PASS Algorithm). Consider a dataset, $D = \{(1, 5.9), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6), (10, 15)\}$ and threshold $\theta = 5°$. The *PASS* approach will first create the set of windows, $W = \{[(1, 5.9), (2, 6)], [(2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6)], [(9, 6), (10, 15)]\}$. The algorithm will then select a subset S, where $S = \{(1, 5.9), (2, 6), (9, 6), (10, 15)\}$ by choosing the first and last points from each window. A tuple set is used to avoid duplicates.

Definition 3.2 (Square Error). Given a time series dataset D and a sample S, we define square error E as the following:

$$E = \sum_{i=1}^{N} (v_i - v_i')^2 \tag{4}$$

Here $N = |D|$ is the length of the original dataset. v_i is the value of D at the time instance t_i, i.e., the i^{th} tuple. v_i' is the value in S, i.e., the sampled dataset at time t_i for the i^{th} tuple. If time t_i is

present in S, then we use the corresponding v_i, otherwise we use linear interpolation to find the value v'_i. For the sample dataset discussed in the Example 3.1, the square error is calculated as 0, and the semantics of the trend will be completely preserved if we use S to make the line chart. The ratio of $\frac{|D|}{|S|} = \frac{10}{4} = 2.5$. So, we have an error of 0 with a threshold $\theta = 5°$.

Example 3.2 (Larger Threshold). For the dataset discussed in the Example 3.1, if we increase the threshold θ to 10°, we will get a smaller sample but with some square error. While creating a window, the tuple $(1, 5.9)$ is merged with the second window, which introduces some error. The new set of windows will be $W = \{[(1, 5.9), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6)], [(9, 6), (10, 15)]\}$. Now, the reduction ratio $\frac{|D|}{|S|}$ will be $\frac{10}{3}$ indicating a smaller sample size.

An interesting property of our approach is that it can form different kinds of windows within a dataset. These windows can be classified into two categories based on the density factor.

Definition 3.3 (Density Factor). The density factor of a window is defined as follows,

$$d = 1 - \frac{P_s}{P_t} \tag{5}$$

Here P_s is the total number of points not selected and P_t is the total number of points present in that window. The value of d is in the range of $[0, 1)$.

Sparse Window. When a window selected by the algorithm has only a few points and the density factor $d \leq 0.5$ then the window is sparse. A window can have as low as two points. This type of window does not help with data reduction in this approach. The window has only two points and all of those points need to be selected to preserve the characteristics of the window.

Dense Window. A dense window is a window where the number of points is more than 2 and density factor $d \geq 0.5$. Our approach can exploit these dense windows to achieve significant data reduction while preserving the semantics.

The density factor d affects the percentage of points selected as sample. The data reduction ratio increases as the density factor d moves closer to 1. The relation of density factor with data reduction ratio is shown in Lemma 3.1.

Lemma 3.1. *If the maximum density factor is d_m then the data reduction ratio can not exceed $\frac{1}{1-d_m}$ i.e. $\frac{|D|}{|S|} \leq \frac{1}{1-d_m}$.*

Proof. Let us assume that all the windows have the same maximum density factor $d_m = 1 - \frac{P_s}{P_t}$. So, $1 - d_m = \frac{P_s}{P_t}$. Hence, the data reduction ratio for this window is $= \frac{P_t}{P_s} = \frac{1}{1-d_m}$. So if all the windows have same d then the data reduction ratio will be the same for all the windows. Hence, the data reduction ratio for this case will be as follows,

$$\frac{|D|}{|S|} = \frac{1}{1 - d_m}. \tag{6}$$

On the other hand, if density factor in some of the windows is less than d_m then our reduction ratio will be $\frac{|D|}{|S|} < \frac{1}{1-d_m}$. Hence, $\frac{|D|}{|S|} \leq \frac{1}{1-d_m}$. □

Next, we discuss the window selection algorithm in the following subsection.

ALGORITHM 1: Window Selection Algorithm

```
1:  procedure SelectWindow (D, θ)
2:      W ← { }
3:      if len(D) ≤ 2 then
4:          W ← W ∪ D
5:          return W
6:      p1, p2 ← D(0), D(1)
7:      Wc ← {p1, p2}
8:      θw ← angle(p1, p2)
9:      pp ← p2
10:     i ← 2
11:     while (i < len(D)) do
12:         pc ← D(i)
13:         θc ← angle(pp, pc)
14:         if |deviation(θw, θc)| ≤ θ then
15:             Wc ← Wc ∪ pc
16:             pp ← pc
17:         else
18:             W ← W ∪ Wc
19:             Wc ← {pp, pc}
20:             θw ← angle(pp, pc)
21:         i ← i + 1
22:     W ← W ∪ Wc
23:     return W
```

3.1 Window Selection Algorithm

In *PASS*, we need to first select the windows (W_i) of a similar threshold (θ) from the dataset (D). Then, we choose points from those windows to get the reduced sample. The window selection algorithm is given in Algorithm 1.

The algorithm takes D and θ as input and produces windows as output when D has more than two points. The algorithm starts by putting the first two points p_1, p_2 into the running window W_c in line 7. Then, the algorithm calculates the angular orientation of the two points θ_w with respect to the x axis in line 8. This angular orientation becomes the angle of the current window W_c. Then, the algorithm iterates over the remaining points in the dataset. The algorithm also calculates the angular orientation θ_c of the current point p_c, and the previous point p_p with respect to the x axis. If the absolute difference of θ_c and θ_w is less than or equal to θ, then it includes the current point p_c in the running window W_c in line 15 and assign p_c to p_p in line 16. Otherwise, if the absolute difference is greater than the threshold θ, then we close the previous window and put that into the collection of windows W in line 18. We start a new running window with the points p_p, p_c, then update the θ_w, and continue this process till the endpoint of the dataset. Finally, we return the collection of windows in line 22.

Example 3.3 (Example of Window Selection). Let us assume that we have the line chart in Figure 2 and the threshold is 30°. We start the first window W_1 with orientation angle $\angle W_1 = 75°$ with respect to the horizontal axis using the points (p_1, p_2). As we scan the next subsequent points, we keep adding the new points in the window W_1 if and only if the deviation of the point p_{i+1} forming line segment $p_i p_{i+1}$ at orientation angle $\angle p_i p_{i+1}$ with respect to the horizontal axis is less than 30°

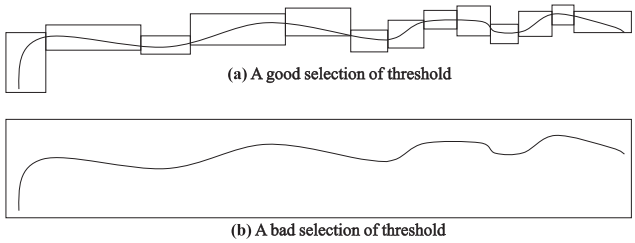


Fig. 3. Effect of threshold in window selection. The image (a) shows a better choice of threshold creating multiple windows. The image (b) shows that selecting a large threshold can result in a single window with better compression but result in a loss of trend information.

i.e., $abs(\angle p_i p_{i+1} - \angle W_1) \leq 30°$. Here, $abs(\angle p_i p_{i+1} - \angle W_1)$ is the deviation of the point p_{i+1} from the current window W_1. As we proceed, we first observe the point p_7, for which the deviation of the line $p_6 p_7$ at the orientation $\angle p_6 p_7$ deviates more than 30° from the current window. So, we begin a new window W_2 starting from p_6 with $\angle W_2 = \angle p_6 p_7$. We continue in this way till the last point p_{29} to form the windows $\{W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8\}$.

The algorithm is a single-pass algorithm with complexity $O(n)$. Here, n is the size of the dataset. The algorithm iterates over all the data points and creates several windows. Extra memory of $O(|D|)$ is used to store the windows. We can reduce this additional memory requirement by selecting points from the windows on the fly and inserting it into the sample dataset.

3.2 Point Selection from Windows

After selecting different windows W_1, W_2, W_3, ..., W_n from the original dataset, we select some points from the window. We have found that when the threshold is small, selecting only the first and last points from each window results in less error compared to competing approaches. To select points, we have to avoid adding repeated points from windows to ensure that we use a tuple set to collect the samples. If the selection of threshold θ is not appropriate, the point selection strategy can cause a higher error. For example, if we use a very large θ so that only a single large window is selected, we might obtain only two points in reduced dataset with huge error.

For example, if we select only the two end points in Figure 3(b), we will lose important details and anomalies incurring large error because here the threshold θ is a bad choice and we select a single window out of the entire dataset even though the trend is not a straight line. For the trend line in Figure 3(a), if we select only the two end points from each window then we will not lose any peak and anomaly. The trend will be preserved, and the resulting sample will have less error. Therefore, if the points are close enough, the proposed technique needs to check the angular orientation much more times. It increases the computation cost. On the other hand, if the points are far away, more errors will be generated.

Efficient point selection from windows is necessary to get the samples out of the original dataset. The number of points selected as sample depends on the windows created using the algorithm. The most efficient point selection strategy is to select the first and the last point from each window. This will preserve the semantics of the trend as well as choose the least number of points given the number of windows selected with the used threshold. But, if the trend is not linear, then a curve fitting strategy can be applied to select points from the curve to minimize the error [55]. So far, we have mainly worked with datasets having mostly linear windows in the trend. The datasets in the experiment have long linear windows at the same orientations. So, we do not incur any loss selecting the first and last points from these windows. If we have selected any other two points

instead of them, we would lose the trends forming different structure than the original chart. Again, including more points also did not add additional benefits as only two points, (x_1, y_1) and (x_2, y_2) are sufficient to describe a straight line. Moreover, adding more points would negatively affect the performance in terms of compression ratio. So, selecting only first and last point from each window was sufficient to achieve high performance in data reduction.

Observation 1. *Given a threshold θ and a trend line data D such that the trend is a straight line, PASS will only select two endpoints and create only a single window. This is the best case scenario.*

The threshold θ plays the key role in PASS. In random sampling, we can specify the percentage and the algorithm will select that percentage of points from the original data. In PASS, θ is the only parameter that determines how many windows will be created and how many points will be selected as a sample out of the original dataset.

Observation 2. *If the threshold θ is less than the minimum deviation of all points from the corresponding current window, then a maximum number of windows will be created, and all the points will be selected as sample. This is the worst-case scenario for PASS.*

Next, we investigate the lower bound of our algorithm in terms of sample size. In the following observation, we show that if our chosen threshold θ is greater than the maximum angular deviation from the respective current windows then only two points will be selected as sample.

Observation 3. *If the threshold θ is greater than the maximum deviation of all points from the corresponding current windows, only one window will be returned by the window selection algorithm and the number of points selected as sample will be* 2.

3.3 Finding Right Threshold

In *PASS*, selecting threshold has an impact on getting lossless visualization from the sample as well as sample size. In Observation 2 and Observation 3, we have presented lower and upper bounds for choosing a threshold. To get an effective sample we need to select a threshold that will lie in between the bounds. First, we want to choose a threshold θ that selects a fixed sample size out of the original dataset. Second, we want to choose a threshold of θ that minimizes the error in the sampled dataset.

3.3.1 Finding a Threshold for the Desired Sample Size. It is often desired to ask for some $x\%$ sample. For example, in random sampling, we can get $x\%$ of the original dataset. The best effort algorithm for selecting a sample with a fixed size is shown in Algorithm 2.

The algorithm takes the original dataset D and the target size of the sample data n. The predefined tolerance τ in the algorithm is used so that the algorithm can be terminated for a sample size within $n - \tau \leq |S| \geq n + \tau$. Here, τ depends on the choice of a user. The algorithm first computes $l = |S|$ using the initial θ. Then, we iteratively refine the threshold. If the sample size l is less than $(n - \tau)$, we decrease the threshold θ and assign a new value $\theta = \theta - \alpha * \theta$ in line 7, where α is a predefined learning rate. If sample size l is less than n, as per the Observation 2 and 3 , we have to decrease the value of θ to increase the number of sample points. So, we decrease the θ using a preset learning rate. If the length of sample l is greater than $n + \tau$ we increase θ and assign a new value $\theta = \theta + \theta * \alpha$ in line 9 as per Observation 2 and 3 ,we have to increase θ to reduce the number of sample points. To ensure termination guarantee, we return Nil if we do not find an acceptable threshold within predefined maximum allowed iterations I.

3.3.2 Finding a Threshold to Minimize the Sample Size for a Given Error Tolerance. To find a threshold that minimizes error, we use a gradient descent approach that iteratively minimizes the error E using a decay rate α. The algorithm reduces the threshold of θ and reduces the number of points selected iteratively until the error falls below an acceptable tolerance ϵ set by the user. The

ALGORITHM 2: Selection of optimal θ to get $|S| = n$

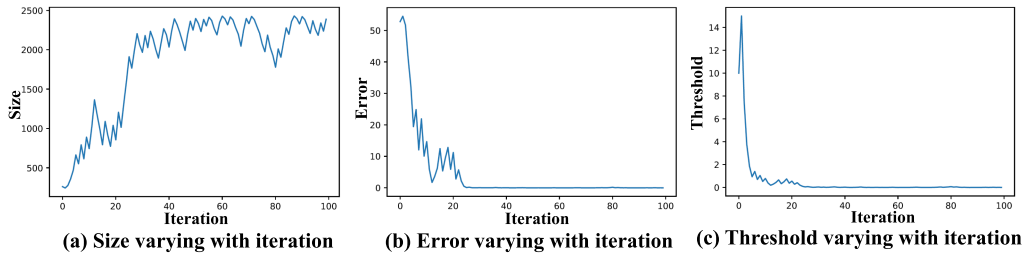
```
1:  procedure GETNSAMPLE (D, n)
2:      θ, S, i ← 0.01, getSample(D, θ), 0
3:      E ← getE(D, S)
4:      l ← len(S)
5:      while true do
6:          if l < n − τ then
7:              θ ← θ − α ∗ θ
8:          else if l > n + τ then
9:              θ ← θ + α ∗ θ
10:         else
11:             break
12:         i ← i + 1
13:         S ← getSample(D, θ)
14:         E ← getE(D, S)
15:         l ← len(S)
16:         if i ≥ I then
17:             return Nil
18:     return θ
```

ALGORITHM 3: Selection of optimal θ to achieve $E \leq \epsilon$

```
1:  procedure OPTMUMTHETA (D, ϵ)
2:      θ, S, i ← 180, getSample(D, θ), 0
3:      E ← getE(D, S)
4:      if θ = 180 || E ≤ ϵ then
5:          return θ
6:      else
7:          while E > ϵ do
8:              θ ← θ − α ∗ θ
9:              i ← i + 1
10:             S ← getSample(D, θ)
11:             E ← getE(D, S)
12:             if i ≥ I then
13:                 return Nil
14:         return θ
```

goal of Algorithm 3 is to return an optimal value of θ so that the error function produces a value less than an acceptable limit of ϵ due to the reduction of original data D to sample data S.

The error function is shown in the Equation (4). The error will be minimum when $|D| = |S|$ and the error will be $E = 0$. It is not required to find the minimum error that returns the sample where $|S| = |D|$. If we plot the square error curve against the sample size, then we will have a lowest point in the curve where the error is the minimum. For $|D| = |S|$ the point in the error curve is the lowest showing $E = 0$. But that is not desired while reducing data. We stop the iteration in our gradient descent algorithm once we see that $E \leq \epsilon$. Here, ϵ is an acceptable level of error. We chose a learning rate α and run the gradient descent approach shown in Algorithm 3 iteratively. Algorithm 3 takes the original dataset D, initial chosen threshold θ, acceptable maximum error limit ϵ, maximum number of iterations I to be used before terminating the algorithm with Nil



(a) Size varying with iteration (b) Error varying with iteration (c) Threshold varying with iteration

Fig. 4. Different gradient descent approaches used for various types of optimizations of the *PASS*.

result and a learning rate α as parameters. In each iteration of the while loop in line 4, if $E > \epsilon$, we decrease θ and assign a new value of $\theta = \theta - \theta * \alpha$ in line 5. If we find $E \leq \epsilon$, then we break the iteration in line 7 as we have found θ that will give the sum of square error less than ϵ. To understand the properties of the proposed optimized algorithm, we have applied it to an artificial dataset generated from a pure sine wave (frequency 20Hz, duration 5s) with 2,500 data points. This dataset has two desirable properties. First, the curvature of sine wave provides variation in the trend. Second, the density of data points is uniform that allows the study of even lower sample sizes.

We run the algorithm to find optimal θ. We do not break the algorithm when error $E \leq \epsilon$, rather we run it for 100 iterations to show how the error and θ varies with iterations in the chart. Figure 4(a-c) shows how the size of sample dataset S, error E and threshold θ varies with iterations, respectively. We see that at around 25 iterations we get all parameters θ, sample size n and error E to a satisfactory level for this case. Then, we see the threshold and error decrease very slightly but the sample size (with sampling rate ~50%) increases almost near the original dataset size, which is 2,500 data points.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate salient properties of *PASS* and compare them with other eight traditional sampling techniques to assess the effectiveness and usefulness of our algorithm. Here, we have experimentally evaluated the visualization quality and data reduction efficiency of *PASS* via state-of-the-art image comparison metrics. We have also assessed the performance of our algorithm with competing approaches using runtime as a metric. We have conducted two user studies to understand the visual preference of *PASS* over the other competing methods. These users use time series data regularly. In this user survey, we have used seven different time series data from different domains and generated visualizations using reduced data by the approaches under comparison using the same sampling rate. We claim that our approach *PASS* reduces the large-scale data significantly with visualization semantics and anomalous behavior of the time series data. We present the experimental evaluation to support our claims in the following subsections. Firstly, we describe the experimental settings in Section 4.1 and then we present experimental results in Section 4.2.

4.1 Experimental Settings

In this section, we describe the experimental settings used in our experimental evaluation. We have described the datasets used for evaluation in Section 4.1.1 and evaluation metrics in Section 4.1.2. Here, we discuss the traditional data reduction comparable techniques in Section 4.1.3. Then, we present the experimental setup of user study and implementation in Section 4.1.4.

Table 1. Datasets

Dataset Name	Short Name	Description	Size (data points)
Taxi Data	Taxi	Number of NYC taxi passengers in 30 min bucket.	3600
Temperature Data	Temp	Monthly temperature in England from 1723 to 1970.	2976
Integrated Surface Data	ISD	Weather observation of different stations (total 14K+) collected since 1901.	63,337
DEBS challenge 2012 dataset	D12	The dataset contains almost 50M tuples (5.94GB)	50,000,000
DEBS challenge 2013 dataset	D13	The dataset contains almost 30M tuples (4.26GB)	30,000,000
DEBS challenge 2014 dataset	D14	The dataset contains more than 4000M tuples (145.82GB)	4,000,000,000
New York Stock Exchange data	NYSE	7500+ stocks, each stock has around 3k tuples, 0.5Mb/stock.	2862

4.1.1 Dataset. For evaluation we use seven different datasets from different domains. The datasets are shown in Table 1: Taxi [37], Temperature (Temp) [23], the **Integrated Surface Data (ISD)** provided by NOAA [46], the New York stock exchange data (NYSE), and DEBS challenge 2012-2014 data (D12, D13,D14) [26, 27, 45]. These datasets have several important properties that make them a good fit for evaluation. First, they are all publicly available that will allow others to reproduce our results. Second, preservation of semantics for these time series visualizations is a crucial need. Third, each dataset contains a lot of anomalous behavior. Fourth, the density factor is from low to moderate for most of the windows in these datasets. Finally, some of these datasets are used for evaluation in recent competing approaches ASAP and M4. To conduct the experiment, we have used the same sampling rate which has been empirically selected for the evaluation of *MSE*, *SSIM*, *DSSIM*, and *PSNR* using all nine techniques to make comparisons. See details in Section 4.2.5.

4.1.2 Evaluation Metrics and Parameters. We have used four metrics to evaluate the visualization quality which are ***MSE* (Mean Square Error)** [29], ***SSIM* (Structural Similarity)** [63], ***DSSIM* (Structural Dissimilarity)** [29], and ***PSNR* (Peak Signal to Noise Ratio)** [29]. We have described in detail these metrics and parameters in this section so that the experimental results can be reproduced.

- ***MSE* (Mean Square Error)**. *MSE* is a simple [29] and commonly used metric to compare the error measure between two images. *MSE* is a good measure to calculate image quality if the images don't have noise [52]. Here, *MSE* is the measure of image quality in terms of luminance at different pixel locations. This metric takes the square difference of luminance at different pixel locations and takes an average of all the squared luminance distances. So, the unit of *MSE* is $pixel^2$. It can show how the images are different when the observer observes them. The formula of *MSE* is as follows,

$$MSE = \frac{1}{wh}\sum_{x=1}^{w}\sum_{y=1}^{h}(I_{x,y}(V_1) - I_{x,y}(V_2))^2 \tag{7}$$

  Here, w is the width or number of rows of image matrix and h is the height or number of columns of image matrix. $I_{x,y}$ is the luminance value of the image at the pixel location (x, y).
- ***SSIM* (Structural Similarity)** [63]. Let X and Y be the original visualization and the distorted visualization, respectively. Then the *SSIM* between them at image location i is

described by the Equation (8):

$$SSIM(X_i, Y_i) = \frac{(2\mu_{X_i}\mu_{Y_i} + c_1)(2\sigma_{X_i}\sigma_{Y_j} + c_2)}{(\mu_{X_i}^2 + \mu_{Y_i}^2 + c_1)(\sigma_{X_i}^2 + \sigma_{Y_i}^2 + c_2)} \tag{8}$$

Here, μ_{X_i} and μ_{X_i} are the means of X_i and Y_i, respectively; $\sigma_{X_i}^2$ and $\sigma_{Y_i}^2$ are the variances of X_i and Y_i, respectively; and $\sigma_{X_i}\sigma_{Y_j}$ is the cross-covariance between X_i and Y_i . Constants c_1 and c_2 are used for numerical stability. We use Python scikit-image [59] to compute the *SSIM* of two images, which provides a widely-used implementation of *SSIM* as a library and it returns a value 1 if two images to be compared are exactly same and return 0 if the images are totally different.

- ***DSSIM* (Structural Dissimilarity)** [29]. It is derived from the *SSIM* using the following Equation (9):

$$DSSIM(X, Y) = \frac{1 - SSIM(X, Y)}{2} \tag{9}$$

This measures the structural differences between the two images. The value of *DSSIM* can be between 0 and 1. *DSSIM* of 0 means the two images have the same structure whereas 1 means the images are totally different.

- ***PSNR* (Peak Signal to Noise Ratio)** [29]. *PSNR* calculates the peak signal-to-noise ratio between two images using Equation (10). The higher the *PSNR*, the better the quality of the reconstructed image. The unit of *PSNR* is **decibel (dB)**.

$$PSNR = 10 \log_{10} \frac{(2^d - 1)^2 WH}{\sum_{i=1}^{W} \sum_{j=1}^{H} (p[i, j] - p'[i, j])^2} \tag{10}$$

4.1.3 Traditional Techniques for Comparison. We have compared *PASS* with some existing state of the art techniques. We compare with state of the art $O(n)$ algorithms aiming to provide loss-less time series visualizations like M4 [29], **piecewise aggregate approximation (PAA)** [32], and ASAP [51]. We have also compared with the other two common sampling techniques, which are **Random Sampling (RS)** and **Stratified Sampling (SS)**, that use randomized techniques to select points randomly. Moreover, line simplification algorithms **Ramer-Douglas-Peucker (RDP)** [60], and **Visvalingam-Whyatt (VW)** algorithm [61] are also used in our experimental evaluations. These two algorithms are respectively of $O(n^2)$ and $O(n \log n)$. We have used open-source implementations of ASAP, PAA, RS, SS, RDP, and VW. We have implemented MinMax and M4 techniques.

4.1.4 Experimental Setup. **Implementation Details.** We have implemented *PASS* and all other techniques in Python and evaluated on a PC equipped with Intel Core i7-7600 processor, 32 GB of RAM, and display resolution of 1920×1080.

We have excluded data loading time from our results and compute the running time for running the algorithms to generate the sample and the time series visualization.

User Study Setup. To assess how different sampling algorithms affect users' ability to identify time series line chart similar to the actual one, we have conducted a large scale user study on Amazon Mechanical Turk inspired by [15, 51]. We comply with the IRB guidelines provided by our institute while conducting the survey. The users surveyed were regular users of time series data and had expertise in understanding the time series visualizations. Users used the web platform provided by Amazon Mechanical Turk which is independent of any devices. Users were not trained for this user study but we have provided guidelines while doing the survey. The sample user study questions are provided in the repository [18]. We have conducted two user studies using Amazon Mechanical Turk for assessing the suitability of *PASS* in the context of preserving semantics and

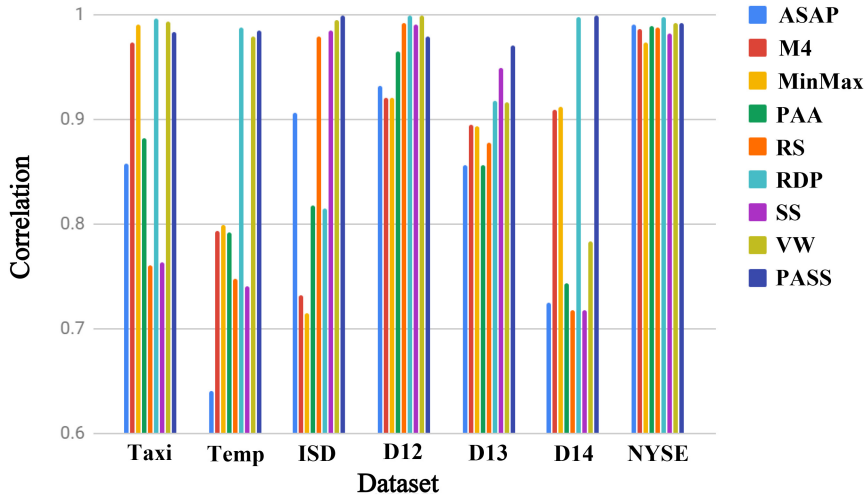


Fig. 5. Measured Correlation for PASS and other techniques with original trend line using different datasets.

anomalous behavior of the time series trend visualization. Visual preference user study and pattern finding user study has been described in details in Section 4.2.3 and Section 4.2.4.

4.2 Experimental Results

In this section, we present the results obtained from the experiment on preserving semantics in Section 4.2.1 and preserving anomaly in Section 4.2.2. Then, we present the user study experiment in Section 4.2.3 and Section 4.2.4. After that, we evaluate the visualization quality of the visualizations created by *PASS* and compare with other approaches in Section 4.2.5. To evaluate that our algorithm is a single-pass algorithm, we present the comparative analysis of runtime of different approaches in Section 4.2.6. Finally, we also assess the effect of threshold θ in the data reduction efficiency of *PASS* in Section 4.2.7. All the experimental results have been shared in the GitHub repository [18].

4.2.1 Experiment on Preserving Semantics. For experimenting on preserving the semantics of line chart after data reduction, we have measured correlation using MATLAB image analysis toolbox for *PASS* and compared with eight other state-of-the-art techniques using seven different datasets as depicted in Figure 5. When the measured correlation with the actual line trend for a specific sampling technique is greater than other techniques, the sampled line trend preserves better semantics of the original line trend in terms of the similar subsequences within that line chart. As shown in Figure 5, *PASS* has a better correlation than other techniques in ISD, D13, D14 datasets compared to all other data reduction techniques. For Temp and NYSE dataset, *PASS* has a comparably similar correlation with the actual line trend like RDP, which is not a linear time algorithm. The measured correlation of *PASS* is not the best in Taxi and D12 dataset according to Figure 5. RDP, VW, MinMax perform better than *PASS* in Taxi dataset. RS, SS, RDP, and VW performed better than *PASS* in D12 dataset.

4.2.2 Experiment on Preserving Anomaly. We have computed the **anomaly preservation score (*APScore*)** using Equation (3) and computed rank for *PASS* compared with other techniques. Table 2 depicts the results. According to Equation (3), the lower *APScore* means more anomaly is preserved in the sampled line trend with respect to the actual line trend. Based on that, the

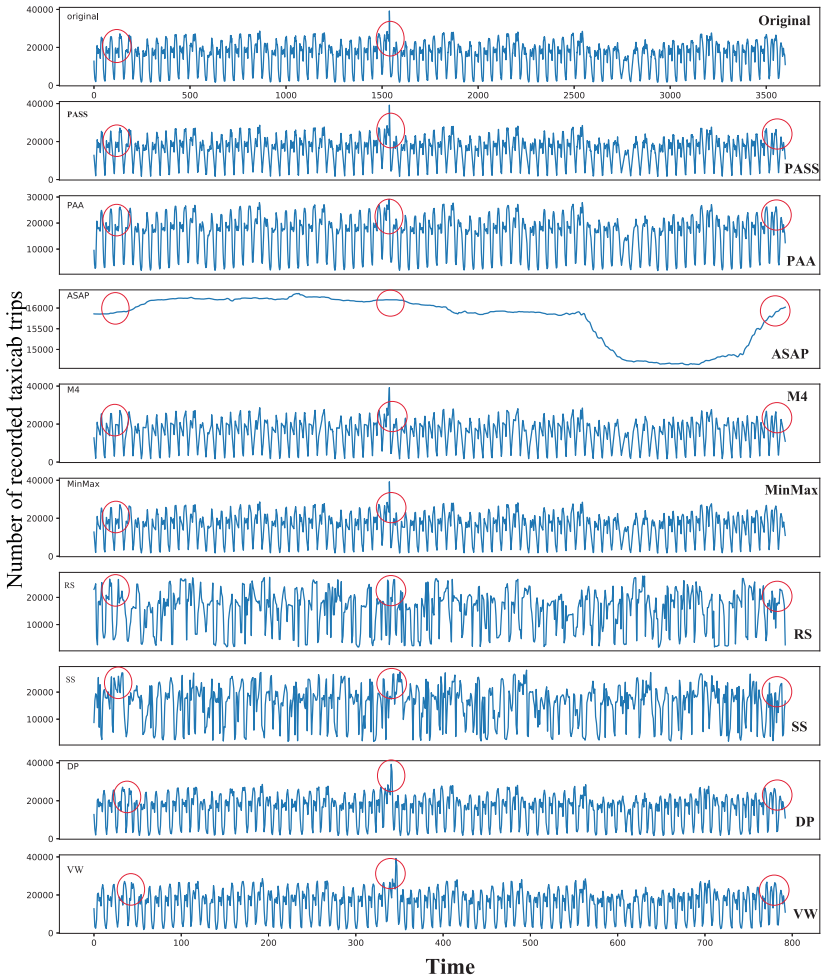


Fig. 6. Experiment with Taxi data.

computed rank of *PASS* is 1 for Temp, ISD, D13, D14 and NYSE dataset, which is observed from the green cells. But computed rank of *PASS* is 3 for Taxi and 5 for D12 dataset. The sampling rate that has been empirically chosen for the Taxi and D12 dataset resulted in a sample size, which could not be achieved by *PASS* at an optimal threshold θ. The threshold θ needed to achieve the same sample size was higher, caused some missing anomalies as illustrated in Figure 6. That is why *PASS* could not perform well in these two datasets compared to the techniques under comparison.

4.2.3 User Study: Visual Preference. **Study Design:** In the user's visual preference study, users are asked to find almost similar visualizations generated using different sampling techniques compared to the actual time series visualization. In this study, we randomize the order in which visualizations are shown. We have also chosen four comparable visualizations after randomizing the subsets to get a balanced sampling across all the nine techniques where *PASS* was the common option. To experiment with this setup, we have divided the interface design into two sets of randomly chosen five techniques at a time where *PASS* was kept common for comparison. The users were

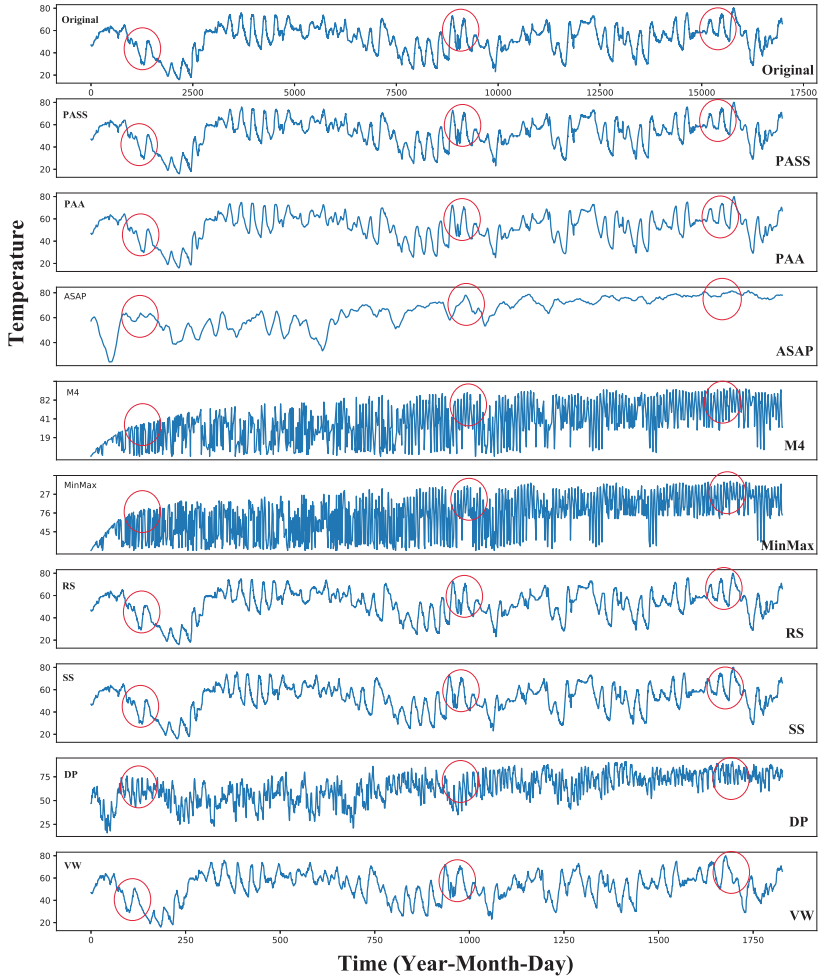


Fig. 7. Experiment with ISD data.

presented with different instances of images for each dataset. For instance, in case of Taxi dataset (Figure 6), PASS and the other eight techniques were presented to the user after dividing into two sets of five techniques chosen randomly at a time to compare with *PASS*. Similarly, Figure 7 depicts all the visualizations generated from the original data and sampled data using all the approaches under comparison for ISD data. We show these two combined sets of visualizations as an example here because our approach performed best for the ISD data and performed at 75% quartile for the Taxi data. In those figures, we have marked the locations where semantics losses are clearer with the red circle for better understanding.

Results and Discussion: In total, 1,750 user responses have been collected in the user preference study. The users carefully observed and used their experience to answer which one is the closest to the original image. Based on the responses, we have plotted a graph showing the percentage of user responses against each dataset for different techniques, as depicted in Figure 8. From the results shown in Figure 8, it is observed that among all the methods *PASS* has sustained marked anomaly of the original trend line for all the datasets except D12 dataset.

Table 2. Comparison Among Traditional Techniques with Experimental Metrics using 7 Datasets

Dataset	Metrics	Techniques								
		PAA	ASAP	M4	MinMax	RS	SS	RDP	VW	PASS
Taxi	*MSE*	2780.07	3562.69	482.08	**177.73**	4955.98	4899.26	11457.59	11447.19	732.91
	SSIM	0.68	0.54	0.93	**0.97**	0.4	0.4	0.74	0.74	0.93
	DSSIM	0.16	0.23	0.04	**0.01**	0.3	0.3	0.13	0.13	0.04
	PSNR	32.91	33.28	35.7	**36.71**	31.3	31.41	35.47	35.52	36
	APScore	19.8765	45.5962	0.5808	**0.0499**	118.7535	116.9833	56.7471	56.6158	0.8756
	Running Time (sec)	0.012	0.008	1.323	1.299	0.008	**0.004**	2.668	0.119	0.2599
	Sampled Data Points	900	977	800	800	**792**	828	800	800	796
Temp	*MSE*	11979.76	9060.89	3691.93	3574.43	5448.92	5784.93	379.4	523.75	**296.08**
	SSIM	0.23	0.37	0.34	0.35	0.36	0.34	**0.92**	0.91	0.91
	DSSIM	0.39	0.31	0.33	0.33	0.32	0.33	**0.04**	0.05	0.05
	PSNR	30.53	30.54	30.73	30.81	30.19	30.08	33.98	**34.07**	33.61
	APScore	665.3621	248.5775	116.6073	109.3858	160.4334	186.6617	0.4855	0.8447	**0.4840**
	Running Time (sec)	62.83	0.0159	1.073	1.0709	**0.00299**	0.0069	3.5279	0.115	0.268
	Sampled Data Points	744	**598**	800	800	744	744	745	745	759
ISD	*MSE*	3827.61	2208.84	6613.49	7179.07	505.44	332.32	12893.18	76.34	**0**
	SSIM	0.6	0.68	0.44	0.44	0.93	0.95	0.46	0.99	**1**
	DSSIM	0.2	0.16	0.28	0.28	0.03	0.02	0.27	0.01	**0**
	PSNR	33.16	36.1	30.72	30.34	38.89	39.24	32.98	42.1	**100**
	APScore	38.4762	14.3969	136.9982	150.5768	0.4192	0.1783	229.4644	0.0183	**0.0000**
	Running Time (sec)	0.02199	0.021	18.471	15.74	**0.007**	0.012	2.668	0.4479	1.2968
	Sampled Data Points	1768	1760	1800	1800	1868	1867	1791	1791	**1759**
D12	*MSE*	11435.25	11729.45	12071.77	12080.79	332.38	343.78	**2.16**	3.87	688.99
	SSIM	0.71	0.65	0.62	0.62	0.96	0.96	**1**	**1**	0.92
	DSSIM	0.15	0.18	0.19	0.19	0.02	0.02	**0**	**0**	0.04
	PSNR	38.08	37.84	36.31	36.3	41.38	40.76	46.7	**47.34**	39.63
	APScore	63.4427	85.8392	101.8841	101.9884	0.1673	0.1757	**0.0000**	**0.0000**	0.7559
	Running Time (sec)	0.1564	0.1404	1.9311	1.7274	**0.1511**	0.1524	1.67	0.1238	0.3195
	Sampled Data Points	2500	**2403**	2439	2440	2500	2500	2529	2501	**2403**
D13	*MSE*	13765.27	13765.27	13717.61	13730.17	3060.98	1131.1	1811.12	1836.94	**650.63**
	SSIM	0.49	0.49	0.51	0.51	0.63	0.83	0.75	0.75	**0.89**
	DSSIM	0.25	0.25	0.25	0.25	0.19	0.08	0.13	0.13	**0.06**
	PSNR	34.15	34.1	34.1	34.08	33.97	36.79	35.53	35.46	**39.53**
	APScore	205.6544	205.9559	197.1941	197.4905	27.1755	2.9634	8.8356	8.9792	**1.1096**
	Running Time (sec)	**0.13129**	1.0653	3.1969	3.02	0.225	0.2612	5.488	0.259	0.4022
	Sampled Data Points	4975	4975	4840	4840	4876	4875	4989	4951	**4809**
D14	*MSE*	16112.6	16786.32	12198.87	12148.99	7712.67	7868.36	25.94	5101.2	**4.47**
	SSIM	0.39	0.38	0.39	0.4	0.47	0.46	0.98	0.62	**0.99**
	DSSIM	0.31	0.31	0.31	0.3	0.27	0.27	0.01	0.19	**0**
	PSNR	30.57	30.15	34.54	34.78	29.54	29.4	44.38	31.16	**53.24**
	APScore	418.9549	454.1991	280.7336	261.9822	149.9893	157.0879	0.0060	50.1692	**0.0000**
	Running Time (sec)	0.8934	**0.033307**	14.6166	14.5019	0.17319	0.191	55.534	4.384	4.6251
	Sampled Data Points	4546	4545	4840	4840	5000	5000	**4178**	5164	4425
NYSE	*MSE*	9143.11	9210.67	449.27	647.71	261.15	386.33	10800.53	162.55	**147.31**
	SSIM	0.81	0.81	0.95	0.91	0.96	0.94	0.77	**0.98**	**0.98**
	DSSIM	0.1	0.1	0.03	0.04	0.02	0.03	0.11	**0.01**	**0.01**
	PSNR	41.99	42.13	43.58	41.62	44.01	43.06	41.32	45.21	**45.61**
	APScore	26.8821	26.9907	0.3256	0.6841	0.1236	0.2863	37.3411	0.0367	**0.0330**
	Running Time (sec)	0.0049	0.168	0.9961	0.81999	**0.00199**	0.003978	1.3528	0.0689	0.27
	Sampled Data Points	573	**477**	599	600	572	572	544	544	541

* Green cells indicate the best values. Blue and Yellow cells indicate the 2nd and 3rd best values, respectively.

To sum up, we have obtained that in the first set, 47.77% of the responses prefer PASS among the other four techniques, and in another set, 45.14% of the users' responses prefer PASS among the other four techniques which make the reported results unbiased and interpretable. To understand how easy it was to find the similar visualization to the original one, we have also recorded the response time. The average response time for different approaches among all the datasets is shown in Figure 9. We have noticed that *PASS* is as easy as other methods to identify patterns and anomalies by the users. So, users can easily distinguish and choose the best line chart drawn using reduced data that resembles the original line chart.

4.2.4 User Study: Pattern Finding. **Study Design:** We have conducted another user study for pattern finding in a time series visualization where users have been asked to justify if the visualization generated from a technique contains similar patterns like the original one. They are also asked to rate the similarity of the charts to the original chart based on the marked patterns using a rating scale between 1 to 5. Thus, we have performed one to one comparison of each of the nine

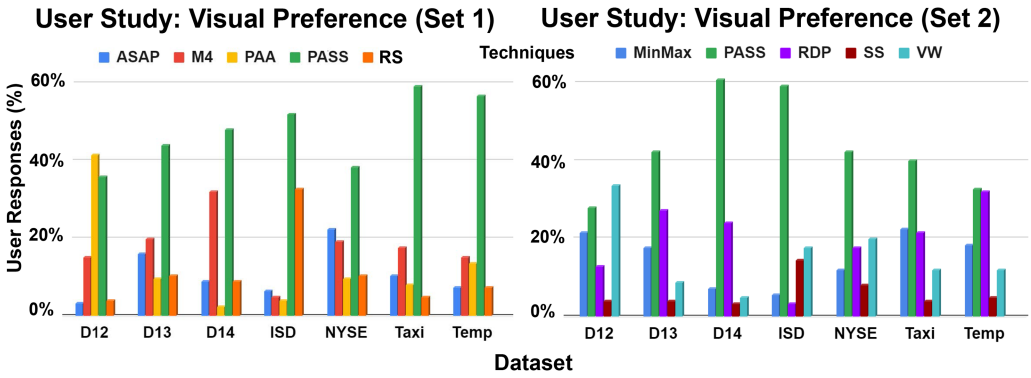


Fig. 8. User's Visual Preference experimental results of the visualizations generated from randomly chosen four techniques at a time for comparing *PASS* using seven datasets.

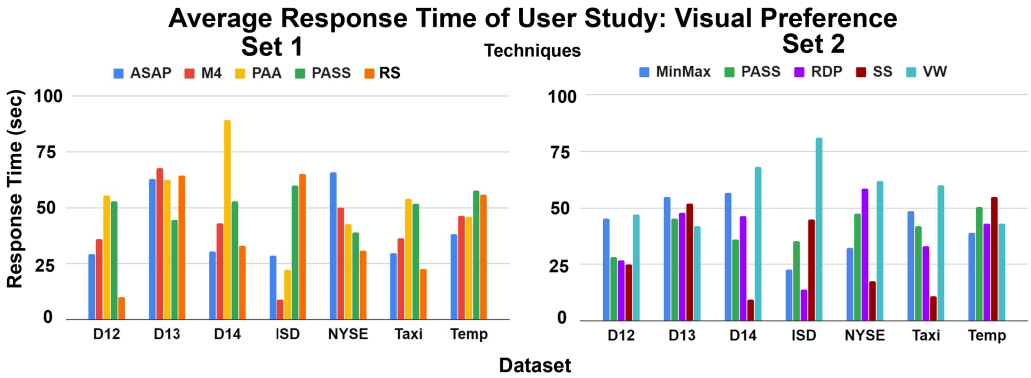


Fig. 9. Average Response Time of visual preference user study of nine techniques using seven datasets.

techniques in a single batch for seven datasets. In total, 1,575 user responses have been collected in the pattern-finding study. A total of 21 different patterns were used in the pattern-finding user study as we have used seven different datasets each with different time series plots containing three randomly marked patterns. All the images with marked patterns used for experiment have been shared in this GitHub repository [18].

Results and Discussion: In this user study, we have observed that 22.13% of the users rated *PASS* as exactly similar to the original trend in terms of pattern-finding, which is the highest among all other comparable techniques. Figure 10 (on Page 19) illustrates the comparison between *PASS* and other techniques in different user rating responses and the average rating obtained for each technique.

4.2.5 Visualization Quality and Data Reduction Efficiency. In this section, we evaluate the visualization quality of *PASS* with other competitive approaches using the metrics *MSE*, *SSIM*, *DSSIM*, and *PSNR*. Throughout the experiment, we have used the same sampling rate for all techniques. The sampling rate in each of the datasets has been empirically selected so that the visualization of the original data and the visualization of the sampled data are $\geq$0.4 *SSIM* different (where 0.4 is the threshold and *SSIM* is the metric for measuring similarity) in the case of the Taxi dataset, and $\geq$0.44 *SSIM* different in the case of the ISD dataset. As we can observe from two example visualizations in Figures 6 and 7, the generated images from sampled data using all the techniques have

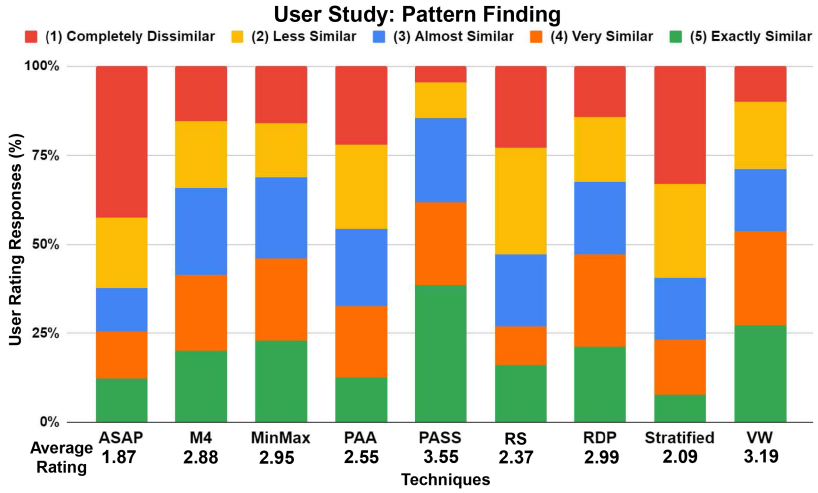


Fig. 10. Pattern Finding experimental results of rating of visualizations from nine techniques using seven datasets.

better visualization quality compared to the original images. In the empirical evaluation, we have obtained a sampling rate of 22%, 25%, 2.7%, 48%, 48%, 4.8%, and 19% for the dataset presented in Table 1 i.e., Taxi, Temp, ISD, D12, D13, D14, and NYSE data, respectively. According to the experimental results depicted in Table 2, in the Taxi dataset, *PASS* stands third in terms of *MSE* and *SSIM*. MinMax and M4 perform slightly better than *PASS* for this dataset in terms of *MSE* and *SSIM*, but in terms of *PSNR*, *PASS* beats M4. In Temp, ISD, D13, D14, and NYSE *PASS* outperforms all the competing approaches in terms of MSE. In the D12 dataset, RDP and VW approaches beat *PASS*. In terms of *SSIM*, *DSSIM*, and *PSNR*, *PASS* outperforms other approaches in most of the cases. In Table 2, the Sampled Data Points row represents the minimum number of points after sampling each dataset for each technique that can regenerate the original time series graph and preserve visualization quality. We can observe from Table 2 that *PASS* can reduce the most data points, preserving visualization quality in the ISD, D12, and D13 datasets compared to the other approaches. In Taxi, D14, and the NYSE datasets, *PASS* is the 2nd best technique to reduce the number of points and sustain the visualization quality. In Temp, D12, and the NYSE datasets, even though ASAP can reduce the most data points, it fails to achieve the best values of *MSE*, *SSIM*, *DSSIM*, and *PSNR*. We observe that in most of the evaluated datasets, *PASS* outperforms other approaches in preserving original semantics and interesting anomalous behaviors in large-scale time series data.

4.2.6 Performance Analysis. In this section, we demonstrate the performance of our algorithm in terms of runtime. Our algorithm is implemented in Python. We have also done a faithful implementation of the M4 and MinMax algorithms. For other algorithms, we have used already publicly available Python libraries. These publicly available algorithms through well-known vendors are optimized through a C++ back-end in most cases. So, these algorithms got favor in runtime comparison. We have seen *PASS* performs well compared to the existing $O(n)$ runtime algorithms M4 and MinMax. *PASS* performs much better than the $O(n^2)$ algorithm RDP, PAA, SS, and VW, which runs faster than some of the cases due to their optimized public API provided by scikit-learn. ASAP performs better than *PASS* for smaller data. But if the data size increases, the runtime of ASAP and *PASS* becomes comparably similar. For D13, *PASS* performed 2X+ faster than ASAP to get the same amount of reduction. We have noticed from Table 2 that *PASS* performs better most of the time in

Table 3. Effect of θ on Data Reduction Using *PASS*

θ	Taxi	ISD	Temp	NYSE	D12	D13	D14
	3600	17000	2976	2862	5000	10000	10000
5	1378	9200	946	105	2271	265	4425
10	1196	9200	888	20	2271	160	4425
15	1117	1759	888	10	2271	110	4425
20	1078	1786	888	7	2271	89	4425
25	1053	204	888	4	2271	68	4425
30	1038	202	888	4	2271	65	4425
35	1025	74	888	4	2271	56	4425
40	1009	17	888	4	2271	50	4425
45	998	10	888	4	2271	41	4425
50	995	10	888	4	2271	38	4425
55	983	2	888	4	2271	26	4425
60	976	2	888	4	2271	20	4425
65	976	2	888	4	2271	17	4425
70	968	2	888	4	2271	14	4425
75	966	2	888	4	2271	8	4425
80	962	2	888	4	2271	8	4425
85	958	2	886	4	2271	2	4425
90	952	2	851	4	2	2	4425

terms of run time compared to M4, MinMax, and ASAP implemented using similar technology. So, our intuition is that if *PASS* is implemented in the optimized coding environment like PAA or VW, then it should perform faster compared to those techniques.

4.2.7 Effect of Threshold on Data Reduction. In this section, we show the effect of threshold θ on data reduction efficiency of *PASS* with Table 3 in which the first row shows the original size of data i.e., total number of data points on which we run the experiment of tuning θ. The following rows show the different reduced sample sizes at 5 degree intervals up to 90 degrees. The maximum value of θ can be 180 degree. To show how increasing the θ results in decreasing sample size, we chose up to 90 degree. This supports the upper and lower bounds of the algorithm explained in Observation 2 and 3 (Section 3.2). In some of the dataset, sample size decreases gradually, because the θ in those dataset change very slowly for a longer period of time. The dataset for which the sample size decreases very fast have dense windows in some few degrees. In some datasets, we notice the size becomes constant after a certain threshold. This is for the case that, they have a number of windows of constant θ at different degrees and the algorithm chose two endpoints from those windows. Then the threshold θ becomes greater than the maximum window angle of all of those windows and hence they cannot reduce further as discussed in Observation 3 (Section 3.2).

5 RELATED WORKS

In this section, we do a literature survey of related works on sampling, data reduction, and strategies for time series data visualization and compare these related works with our proposed approach.

Random sampling and stratified sampling for data reduction. Reduction of data size for visualization to enable effective exploratory data analysis is a well-known problem [5]. A popular approach for reducing dataset size is stratified sampling [10]. But random sampling is not well suited for visualization of scatter plot and map plot [41]. Park et al. [48] have proposed visualization aware sampling using random sampling. Random sampling, though used widely, can lose important outliers [42] or anomalies, which cause loss of semantics totally in time series data visualization in the line chart. Longbo et al. discussed the problems of random sampling from time-based sliding windows over-weighted streaming data. There are different variants of random sampling algorithm for computing quantiles [19], distinct counts [17], and reservoir sampling [1]. Reservoir sampling [1] is a variant of random sampling to preserve random samples online. Some

researchers have used stratified sampling for reducing original data. In stratified sampling, data is divided into disjoint subgroups [41], and then samples are taken from those subgroups. It selects strata depending on some prior information and then selects more points randomly from that strata [10]. Although these approaches have made significant advances in data reduction methods, the requirement of preserving trend semantics and anomaly in large time series data has still been unnoticed.

Relational database based approach. M4 [29] uses SQL to rewrite visualization queries to generate images of lines by reducing time series. They split the chart into multiple smaller time chunks and select first, last, highest and lowest data tuples from each smaller chunk. Though they achieve very competitive visualization quality, there can be higher semantic loss when observed over a small region using interactive visualization library for large dataset. In our approach, we not only get higher visualization quality when visualized as a static image, but we also provide data analysts option to zoom over different regions and study behavior, anomalies with higher semantic preserving guarantee. There has also been competitive work on data reduction for compact data visualizations [7]. Again, line simplification strategy has been used to reduce the number of data points. A research study [16] uses **perceptually important points (PIP)** to reduce the amount of data. But, the complexity of these approaches are $O(n^2)$ and visualization quality with preserving semantics is also not addressed. Recent research [54] has explored a few algorithms for downsampling data to produce visual representation. In particular, Mode-Median-Bucket, Min-Std-Error-Bucket, Longest-Line-Bucket, Largest-Triangle-One-Bucket, Largest-Triangle-Three-Buckets, Largest-Triangle-Dynamic algorithms have been discussed with comparison against each other in terms of visual characteristics of downsampled line charts using survey and comparison matrix. However, that study has not considered the notion of preserving semantics and anomaly of large time series data formally like *PASS* and performed experimental evaluation with metrics obtained from prior work [29, 62, 63] and also made comparison with state-of-the-art techniques.

Modified sampling strategies for specific task. Research on estimating distinct values using sampling [17, 21] exists as well. There are also sampling strategies to find samples from patterns in subgraphs described in a research study [2]. In a distributed search over hidden web hierarchical database, sampling is used [24]. Detecting effective change in data is done using sampling [12]. These approaches are not suitable for time series data reduction without loss of semantics.

Time series summarization. There are some relevant time series summarization techniques, especially **PLA (Piecewise Linear Approximation)** [34], **PAA (Piecewise Aggregate Approximation)** [32], **APCA (Adaptive Piecewise Approximation)** [33], and **SAX (Symbolic Aggregate approximation)** [38] time series data. Among them, we have compared PASS with PAA and got better performance in aspects of both visualization and user preference study. Again, PLA has been a classic problem in data compression and signal tracking. APCA is used for faster approximate searching on the same index structure. Moreover, SAX is a symbolic representation for time series that allows for dimensionality reduction and indexing with a lower-bounding distance measure. Rahman et al. [50] incrementally improve visualizations of line chart by selecting better samples. In that work, online sampling-based schemes have been used to generate approximations that use as few samples as possible for trend-line visualizations. None of these techniques ensure semantics and anomaly preservation resembling original structure by sampling, which is the primary goal of *PASS*.

Change point detection in time series data. There has been a significant research study [20] for detecting the time points at which behavior change occurs utilizing the change-point detection problem in statistics. In that study, techniques have been proposed for both the batch and incremental versions of this sort of problem. In that study, although a comparison of methods

have been performed with visual change point detection by humans, a large scale user study and experiments with image comparison metrics are missing. Another study [31] consists of a geometric based technique for linear models estimation using time-series data and comparative analyses using different datasets. Moreover, Liu et al. [40] have utilized a relative divergence measure to apply change-point detection in time-series data. None of the approaches points out the necessity of preserving the anomalous behavior of time series data after sampling.

Visual exploration tools for time series data. A significant research study on data science [47] illustrates a comprehensive set of tools e.g., Zenvisage [53] for searching line chart patterns, FastMatch [44] for obtaining the histogram visualizations interactively most similar to a user-specified target, IncVisage [50] for trendline visualization, etc. Rahman et al. incrementally improve visualizations of line chart by selecting better samples in the IncVisage tool [50]. Although in that work, online sampling-based schemes have been utilized to generate approximations that use as few samples as possible for trendline visualizations, comparison between visualizations before and after sampling in the aspect of preserving a similar structure of trend line has not been considered. All of these recent research studies encourage us to focus on an unexplored topic of time series data sampling through which users can obtain from the sampled time series data almost similar visualization to the actual time series data after sampling.

Trajectory simplification. [39] presents an aggressive one-pass error bounded trajectory simplification algorithm by interpolating new data points into a trajectory under certain conditions. Long et al. [43] proposed a min-error problem with exact algorithms for trajectory simplification preserving direction. In these research studies, notion of preserving the semantics and anomalous behavior of trend line with less error and higher visualization quality has not been addressed.

6 CONCLUSION AND FUTURE WORK

Three of the key challenges for visualizing large scale time series data are: to reduce the data, to preserve the semantics of trends in terms of similar subsequences among the trends, and to preserve anomalous behavior. We have presented *PASS*, a linear time data sampling strategy that meets all three of those challenges. Our evaluation using seven large datasets shows that *PASS* performs well compared to existing approaches in improving visualization quality as measured by *MSE*, *SSIM*, *DSSIM* and *PSNR*. Our experimental study shows that users prefer visualization produced by *PASS* compared to others. Our future work would focus on examining the performance of *PASS* on data with non-linear trends as well as in the downstream application e.g., anomaly detection. We would also explore other aspects of semantics such as time warping, frequency analysis, etc., with similar subsequences and find out whether those aspects of semantics are also preserved using *PASS*. Eventually, *PASS* paves the way for crafting semantics and anomaly preserving sampling techniques for other data visualization charts e.g., scatter plot, heat map, histogram, etc. In future work, we could explore several research directions. If graph analyses are given as code [56–58], can we apply similar data reduction strategy for graphs in general? Focusing on the similar kind of goal of collective program analysis [56, 57] as well as *PASS* in terms of reducing the data that is sent to downstream analysis and using the semantics of task that is conducted during downstream analysis to reduce the data, can we determine task-dependent program similarity for finding analogous programs in the realm of data science? Another goal might be to realize *PASS* as part of the shared infrastructure such as *Boa* [3, 6, 13, 14, 25] and understand performance improvements and computation savings that might result from it.

REFERENCES

[1] Charu C. Aggarwal. 2006. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd International Conference on Very Large Data Bases*. VLDB Endowment, 607–618.

[2] Mohammad Al Hasan and Mohammed J. Zaki. 2009. Output space sampling for graph patterns. *Proceedings of the VLDB Endowment* 2, 1 (2009), 730–741.
[3] Hamid Bagheri, Usha Muppirala, Rick E. Masonbrink, Andrew J. Severin, and Hridesh Rajan. 2019. Shared data science infrastructure for genomics data. *BMC Bioinformatics* 20, 1 (2019), 1–13.
[4] Mike Barnett, Badrish Chandramouli, Robert DeLine, Steven Drucker, Danyel Fisher, Jonathan Goldstein, Patrick Morrison, and John Platt. 2013. Stat!: An interactive analytics environment for big data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 1013–1016.
[5] Leilani Battle, Michael Stonebraker, and Remco Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton. In *Big Data, 2013 IEEE International Conference on*. IEEE, 1–8.
[6] Sumon Biswas, Md Johirul Islam, Yijia Huang, and Hridesh Rajan. 2019. Boa meets Python: A Boa dataset of data science software in Python language. In *Proceedings of the 16th International Conference on Mining Software Repositories* (Montreal, Quebec, Canada) *(MSR'19)*. IEEE Press, 577–581. https://doi.org/10.1109/MSR.2019.00086
[7] Mihai Budiu, Parikshit Gopalan, Lalith Suresh, Udi Wieder, Han Kruiger, and Marcos K. Aguilera. 2019. Hillview: A trillion-cell spreadsheet for big data. *Proc. VLDB Endow.* 12, 11 (July 2019), 1442–1457. https://doi.org/10.14778/3342263.3342279
[8] Lei Cao, Wenbo Tao, Sungtae An, Jing Jin, Yizhou Yan, Xiaoyu Liu, Wendong Ge, Adam Sah, Leilani Battle, Jimeng Sun, Remco Chang, Brandon Westover, Samuel Madden, and Michael Stonebraker. 2019. Smile: A system to support machine learning on EEG data at scale. *Proc. VLDB Endow.* 12, 12 (Aug. 2019), 2230–2241. https://doi.org/10.14778/3352063.3352138
[9] Badrish Chandramouli, Jonathan Goldstein, and Songyun Duan. 2012. Temporal analytics on big data for web advertising. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 90–101.
[10] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)* 32, 2 (2007), 9.
[11] C. L. Philip Chen and Chun-Yang Zhang. 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences* 275 (2014), 314–347.
[12] Junghoo Cho and Alexandros Ntoulas. 2002. Effective change detection using sampling. In *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB Endowment, 514–525.
[13] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen. 2013. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *Proceedings of the 2013 International Conference on Software Engineering* (San Francisco, CA, USA) *(ICSE'13)*. IEEE Press, 422–431.
[14] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen. 2015. Boa: Ultra-large-scale software repository and source-code mining. *ACM Trans. Softw. Eng. Methodol.* 25, 1, Article 7 (Dec 2015), 34 pages. https://doi.org/10.1145/2803171
[15] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 61–72.
[16] Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. 2008. Representing financial time series based on data point importance. *Eng. Appl. Artif. Intell.* 21, 2 (March 2008), 277–300. https://doi.org/10.1016/j.engappai.2007.04.009
[17] Phillip B. Gibbons. 2001. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*, Vol. 1. 541–550.
[18] GitHub Repository of PASS. 2020. Experiment codes and the results of Semantics and Anomaly Preserving Sampling Strategy (PASS) for Large-Scale Time Series Data. https://github.com/shibbirtanvin/pass.
[19] Michael Greenwald and Sanjeev Khanna. 2001. Space-efficient online computation of quantile summaries. In *ACM SIGMOD Record*, Vol. 30. ACM, 58–66.
[20] Valery Guralnik and Jaideep Srivastava. 1999. Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, California, USA) *(KDD'99)*. Association for Computing Machinery, New York, NY, USA, 33–42. https://doi.org/10.1145/312129.312190
[21] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Lynne Stokes. 1995. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, Vol. 95. 311–322.
[22] Xi He, Eric H. Y. Lau, Peng Wu, Xilong Deng, Jian Wang, Xinxin Hao, Yiu Chung Lau, Jessica Y. Wong, Yujuan Guan, Xinghua Tan, et al. 2020. Temporal dynamics in viral shedding and transmissibility of COVID-19. *Nature Medicine* (2020), 1–4.
[23] Rob J. Hyndman and M. Akram. 2010. Time series data library. Available from Internet: http://robjhyndman.com/TSDL. (2010).
[24] Panagiotis G. Ipeirotis and Luis Gravano. 2002. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB Endowment, 394–405.

[25] Md Johirul Islam, Anuj Sharma, and Hridesh Rajan. 2019. A cyberinfrastructure for big data transportation engineering. *Journal of Big Data Analytics in Transportation* 1, 1 (2019), 83–94.
[26] Zbigniew Jerzak, Thomas Heinze, Matthias Fehr, Daniel Gröber, Raik Hartung, and Nenad Stojanovic. 2012. The DEBS 2012 grand challenge. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. ACM, 393–398.
[27] Zbigniew Jerzak and Holger Ziekow. 2014. The DEBS 2014 grand challenge. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems* (Mumbai, India) *(DEBS'14)*. ACM, New York, NY, USA, 266–269. https://doi.org/10.1145/2611286.2611333
[28] E. Joy and D. Paris. 1972. Spatial sampling and filtering in near-field measurements. *IEEE Transactions on Antennas and Propagation* 20, 3 (1972), 253–261.
[29] Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. 2014. M4: A visualization-oriented time series data aggregation. *Proceedings of the VLDB Endowment* 7, 10 (2014), 797–808.
[30] Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. 2016. VDDA: Automatic visualization-driven data aggregation in relational databases. *The VLDB Journal–The International Journal on Very Large Data Bases* 25, 1 (2016), 53–77.
[31] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. 2007. Change-point detection in time-series data based on subspace identification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 559–564.
[32] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems* 3, 3 (2001), 263–286.
[33] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record* 30, 2 (2001), 151–162.
[34] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2004. Segmenting time series: A survey and novel approach. In *Data Mining in Time Series Databases*. World Scientific, 1–21.
[35] Eamonn Keogh and Shruti Kasetty. 2003. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery* 7, 4 (2003), 349–371.
[36] Albert Kim, Eric Blais, Aditya Parameswaran, Piotr Indyk, Sam Madden, and Ronitt Rubinfeld. 2015. Rapid sampling for visualizations with ordering guarantees. *Proceedings of the VLDB Endowment* 8, 5 (2015), 521–532.
[37] Alexander Lavin and Subutai Ahmad. 2015. Evaluating real-time anomaly detection algorithms–the Numenta anomaly benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 38–44.
[38] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15, 2 (2007), 107–144.
[39] Xuelian Lin, Shuai Ma, Han Zhang, Tianyu Wo, and Jinpeng Huai. 2017. One-pass error bounded trajectory simplification. *Proceedings of the VLDB Endowment* 10, 7 (2017), 841–852.
[40] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. 2013. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks* 43 (2013), 72–83.
[41] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. imMens: Real-time visual querying of big data. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 421–430.
[42] Sharon Lohr. 2009. *Sampling: Design and Analysis*. Nelson Education.
[43] Cheng Long, Raymond Chi-Wing Wong, and HV Jagadish. 2014. Trajectory simplification: On minimizing the direction-based error. *Proceedings of the VLDB Endowment* 8, 1 (2014), 49–60.
[44] Stephen Macke, Yiming Zhang, Silu Huang, and Aditya Parameswaran. 2018. Adaptive sampling for rapidly matching histograms. *Proc. VLDB Endow.* 11, 10 (June 2018), 1262–1275. https://doi.org/10.14778/3231751.3231753
[45] Christopher Mutschler, Holger Ziekow, and Zbigniew Jerzak. 2013. The DEBS 2013 grand challenge. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*. ACM, 289–294.
[46] National Oceanic and Atmospheric Administration (NOAA). 2017. Integrated Surface Data. ftp://ftp.ncdc.noaa.gov/pub/data/noaa/.
[47] Aditya Parameswaran. 2019. Enabling data science for the majority. *Proc. VLDB Endow.* 12, 12 (Aug. 2019), 2309–2322. https://doi.org/10.14778/3352063.3352148
[48] Yongjoo Park, Michael Cafarella, and Barzan Mozafari. 2016. Visualization-aware sampling for very large databases. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 755–766.
[49] Fotios Petropoulos and Spyros Makridakis. 2020. Forecasting the novel Coronavirus COVID-19. *PLOS ONE* 15, 3 (03 2020), 1–8. https://doi.org/10.1371/journal.pone.0231236
[50] Sajjadur Rahman, Maryam Aliakbarpour, Ha Kyung Kong, Eric Blais, Karrie Karahalios, Aditya Parameswaran, and Ronitt Rubinfield. 2017. I've seen enough: Incrementally improving visualizations to support rapid decision making. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1262–1273.
[51] Kexin Rong and Peter Bailis. 2017. ASAP: Prioritizing attention via time series smoothing. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1358–1369.

[52] David Salomon. 2004. *Data Compression: The Complete Reference*. Springer Science & Business Media.
[53] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proc. VLDB Endow.* 10, 4 (Nov. 2016), 457–468. https://doi.org/10.14778/3025111.3025126
[54] Sveinn Steinarsson. 2013. *Downsampling Time Series for Visual Representation*. Master's thesis. University of Iceland.
[55] Tegoeh Tjahjowidodo et al. 2017. A direct method to solve optimal knots of B-spline curves: An application for non-uniform B-spline curves fitting. *PloS one* 12, 3 (2017), e0173857.
[56] Ganesha Upadhyaya and Hridesh Rajan. 2017. On accelerating ultra-large-scale mining. In *Proceedings of the 39th International Conference on Software Engineering: New Ideas and Emerging Results Track* (Buenos Aires, Argentina) *(ICSE-NIER'17)*. IEEE Press, 39–42. https://doi.org/10.1109/ICSE-NIER.2017.11
[57] Ganesha Upadhyaya and Hridesh Rajan. 2018. Collective program analysis. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE'18)*. Association for Computing Machinery, New York, NY, USA, 620–631. https://doi.org/10.1145/3180155.3180252
[58] Ganesha Upadhyaya and Hridesh Rajan. 2018. On accelerating source code analysis at massive scale. *IEEE Transactions on Software Engineering* 44, 7 (2018), 669–688. https://doi.org/10.1109/TSE.2018.2828848
[59] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. 2014. scikit-image: Image processing in Python. *PeerJ* 2 (6 2014), e453. https://doi.org/10.7717/peerj.453
[60] Mahes Visvalingam and J. Duncan Whyatt. 1990. The Douglas-Peucker algorithm for line simplification: Re-evaluation through visualization. In *Computer Graphics Forum*, Vol. 9. Wiley Online Library, 213–225.
[61] Maheswari Visvalingam and James D. Whyatt. 1993. Line generalisation by repeated elimination of points. *The Cartographic Journal* 30, 1 (1993), 46–51.
[62] Peng Wang, Haixun Wang, and Wei Wang. 2011. Finding semantics in time series. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 385–396.
[63] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

Received June 2021; revised November 2021; accepted January 2022