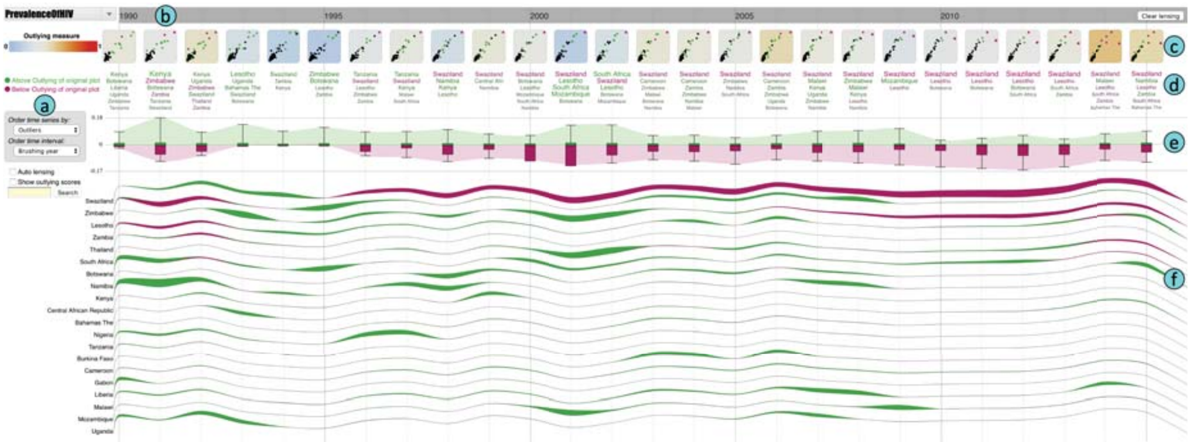
Outliagnostics: Visualizing Temporal Discrepancy in Outlying Signatures of Data Entries

Vung Pham[*] Tommy Dang[†]

Computer Science Department, Texas Tech University



Order time series by: Outliers

Order time interval: Brushing year

□ Auto lensing

□ Show outlying scores

Search

Figure 1: Visualizing the prevalence of HIV (male vs. female) using *Outliagnostics*: (a) The control panel, (b) the lensing area, (c) the scatterplot series, (d) the top countries clouds, (e) the customized outlying boxplots, and (f) the country outlying profiles.

Abstract

This paper presents an approach to analyzing two-dimensional temporal datasets focusing on identifying observations that are significant in calculating the outliers of a scatterplot. We also propose a prototype, called *Outliagnostics*, to guide users when interactively exploring abnormalities in large time series. Instead of focusing on detecting outliers at each time point, we monitor and display the discrepant temporal signatures of each data entry concerning the overall distributions. Our prototype is designed to handle these tasks in parallel to improve performance. To highlight the benefits and performance of our approach, we illustrate and validate the use of *Outliagnostics* on real-world datasets of various sizes in different parallelism configurations. This work also discusses how to extend these ideas to handle time series with a higher number of dimensions and provides a prototype for this type of datasets.

1 Introduction

Analyzing outliers is one of the most fundamental research areas in the field of statistics. An *outlier* is an observation that appears to deviate significantly from the other observations in the sample [20]. Identification of potential outliers is important in many applications. Outliers may indicate errors. For example, the data may have been collected mistakenly, or an experiment may not have been set up and/or executed accurately. Outliers may be extreme cases in a distribution which are particularly interesting and important to be located.

On the other hand, an *inlier* is a value that lies in the interior of a statistical distribution and practically impossible to identify [21], but in the multivariate case, thanks to interrelationships between variables, values can be identified that are observed to be more central in a distribution but would be expected to be more outlying [16]. The term "inlier" that we use in this paper is slightly different. We define inliers as observations that lie in the interior of statistical distribution, and their absences allow identifying outliers easier or possible. In other words, let A and B are two genuine observations in a distribution, but when we remove A, B now becomes an outlier. In this case, A is considered as an inlier. We use this *leave-one-out* approach to measuring the significance (for both inlier and outlier) of individual data points in computing outlying as a whole. In particular, we use the Tukey outlier detection model that leverages the Interquartile range (IQR) to detect outliers in a scatterplot base on the edge lengths of the Minimum Spanning Tree (MST). The leave-one-out approach is computationally intensive, but with the use of parallel computing and binning, the time complexity of our approach is near-linear with respect to the size of a dataset. Our leave-one-out approach is "selective" since we only leave out singleton bins because removing an observation from a dense bin will not affect the outlying scores. Our contributions in this paper are:

- We present an approach for measuring contributions of data points in a scatterplot outlying measure based on the "selective" leave-one-out cross-validation idea. The leave-one-out strategy is applied parallelly over the entire time series to formulate the outlying signatures of individual instance in the data.
- We propose a prototype, *Outliagnostics*, to guide users on interactively exploring high dimensional datasets focusing on outliers and inliers. The visual interface supports a full range of interactions, such as lensing, brushing and linking, and filtering. The interactions and the visual interfaces are non-blocking via multithreading.
- We highlight the benefits of our approach by using *Outliagnostics* on real-world datasets. We conduct an informal study with

[*] e-mail: vung.pham@ttu.edu

[†] e-mail: tommy.dang@ttu.edu

IEEE VDS 2019
20 October, Vancouver, BC, Canada
978-1-7281-5047-5/19/$31.00 ©2019 IEEE

three industry experts on real-time monitoring and detecting unusual events in High-Performance Computing center. We also present a quantitative test to evaluate the feasibility of handling large datasets on different parallelism configurations.

The paper is structured as follows: We describe related work in the following section. Then we introduce our *Outliagnostics* prototype and illustrate it on real datasets. We present use cases and test results on running times with different parallelism settings in the Experiments section. Finally, we conclude our work in the last section.

2 Related Work

In this section, we do not attempt to survey the full range of currently available methods. Instead, we focus on the most related techniques in discovering multivariate outliers. In particular, we review MST outliers in Section 2.1 and other approaches in Section 2.2.

2.1 The Box Plot Rule for MST outliers

We use John Tukey's method of leveraging the Interquartile range (IQR) to detect outliers in a dataset [6]. This method is applicable to most ranges and can be used to detect multidimensional outliers since it is not dependent on distributional assumptions. It also ignores the statistical mean and standard deviation, making it resistant to being influenced by the extreme values in the range. The interquartile range is defined as:

$$IQR = 3^{rd}\ quartile\ value - 1^{st}\ quartile\ value \quad (1)$$

Tukey defines the upper and lower bounds of acceptable data as:

$$upperbound = 3^{rd}\ quartile\ value + IQR * factor \quad (2)$$

$$lowerbound = 1^{st}\ quartile\ value - IQR * factor \quad (3)$$

where the $factor$ is set to 1.5. There does not seem to be any statistically-driven reason Tukey uses 1.5 as a hard basis for his method (we also use 1.5 as the default setting in our method). A larger number (such as 3) could be used to identify the "extreme" outliers. Values above the upper bound or below the lower bound are considered as outliers [38]. Since we are looking for observations that are visually deviated from the other observations in a scatterplot, we only use *upperbound* in our outlier detection algorithm. Outliers are identified by the Box Plot rule (as described above) on the MST lengths. The outlying measures the proportion of the total edge length of the minimum spanning tree accounted for by the total length of outlying edges.

$$c_{outlying} = length(MST_{outliers})/length(MST) \quad (4)$$

2.2 Other Outlier detection methods

There are many survey papers and excellent books on outlier detection written by statisticians [5, 17, 22] and computer scientists [2, 13]. Here we focus on multivariate outlier detection techniques. Rohlf [36] proposes a method of detecting outliers in multivariate data by testing the largest edge of the MST with an assumption that these edges follow a gamma distribution. Similar methods based on the MST have been proposed [30], but they suffer the problem when variates are correlated [12]. Nysia et al. [15] use an iterative leave-one-out approach for outlier detection in RNA-Seq Data, but it is more about improving accuracy rather than reducing computation expense. Takuro and Akihiro [29] propose an outlier detection method based on leave-one-out density using binary decision diagrams to reduce the computation expenses.

Another popular approach to detect multivariate outlier is based on clustering [41]. Pamula et al. [32] apply *k-means* clustering algorithm to divide the data set into clusters. The point which is lying near the centroid of the cluster is not a probable candidate for an outlier, and we can prune out such points from each cluster. Next, we calculate a distance-based outlier score for remaining points. Based on the outlier score, we declare the top *n* points with the highest score as outliers. Jiang et al. [26] propose a two-phase clustering algorithm for outliers detection using a heuristic "if one new input pattern is far enough away from all clusters' centers, then assign it as a new cluster center". In the first phase, the traditional *k-means* algorithm groups data points in the same cluster which may be all outliers or all *non-outliers*. In the second phase, an MST is constructed, and then the longest edge of this MST is removed. Data points in the small clusters (the subtree with less number of nodes) are regarded as outliers. However, most clustering-based outlier detection algorithms do not scale well to larger datasets due to the computations needed to compute cluster iteratively [27].

To deal with the curse of dimensionality, Wilkinson [37] recently proposed an algorithm, called *hdoutliers*, for detecting multidimensional outliers. The algorithm is designed to be paired with visualization methods that can help an analyst explore unusual features in data. The paper also presents a thorough survey with examples on different types of outliers, such as time series outliers, ipsative outliers, text outliers, graph outliers, geographic outliers, and *Scagnostics* outliers. However, none of the above approaches try to detect the temporal behaviors of each data entry with respect to the overall distributions which is significant to capture in many application domains such as terrorism or real-time monitoring health status of high-performance computing systems. Our proposed approach tries to capture these unusual behaviors (genuine observations in the previous time points can suddenly become abnormalities) using leave-one-out. To overcome the time complexity, we use binning (only leave the bins with single element out) and parallel computing.

There are several works using visual interfaces and interactions to explore and validate the outlying data. To name a few, EnsembLens [40] applies ensemble analysis; Viola [10] is based on Canonical Polyadic (CP) decomposition methods with tensor-based anomaly analysis algorithm; TargetVue [11] uses TLOF [8]; Rclens adopts active learning algorithm to identify rare category; CVExplorer [33], MTDES [35], and TimeMatrix [14] use visual overview with supported interactions for discovery and exploratory of data patterns. Zhang et al. [42] also provided a good survey of visualization for network anomalies. Belonging to this class, our work equips users with interconnected views and interactions to explore and validate the significance of individual entries in the overall distribution.

3 Design Decisions

Our proposed approach works with scatterplots contain single outlier or multiple outliers. In case of a single outlier, by leaving the only outlier out, the outlying measure reduces significantly, so it is relatively easy to detect. Our method proves its usefulness in the case of multiple outliers (this is common in many real-world datasets), which are subject to masking and swamping effects [1].

Masking effect: One outlier masks the second outlier if the second outlier can be considered as an outlier only by itself, but not in the presence of the first outlier [24]. In other words, after the deletion of the first outlier, the second instance emerges as an outlier.

Swamping effect: One outlier swamps the second observation if the latter can be considered as an outlier only under the presence of the first one [23]. After the deletion of the first outlier, the second observation becomes a *non-outlying* observation.

Our approach highlights both masking and swamping effects. In general, we want to detect the observations (could be outliers or inliers) which have significant contributions to the outlying computation of a pairwise projection. At the same time, we want to avoid the side effects introduced by removing an observation from the original scatterplot. The next section starts with the design motivations behind selecting to work on 2D projections and other factors that are sensitive to our choice of leave-one-out.

3.1 Motivations: Why 2D?

In many cases, multivariable data points appear to be genuine observations when each variable is considered independently [24]. However, a 2D projection may reveal a very different story [25]. Figure 2 shows an example of various cases of bivariate outliers. One might argue that bivariate outliers in Figure 2(a) are detectable in the marginal distributions on *x* axis (Pakistan), *y* axis (India), or both (China). Nevertheless, it would be hard to refute that Netflix (NFLX) in Figure 2(b) can be possibly detected as an outlier only when multivariate analysis is performed. More obviously in Figure 2(c) when considering separately with respect to the spread of values along the *Life expectancy of Female* and *Life expectancy of Male* axes, Iraq, Iran, and El Salvador fall close to the center of the univariate distributions. The Iran-Iraq war and the Salvadoran Civil war in 1982 account for this shortage on the *Life expectancy of Male* since men were needed for the wars. Thus, the test for outliers must take into account the relationships between these two variables, which in this case appear abnormal (below the diagonal of the scatterplot). Also, this work focuses on bivariate *outlier/inlier* detection, but it could be generalized to work with multivariate cases.

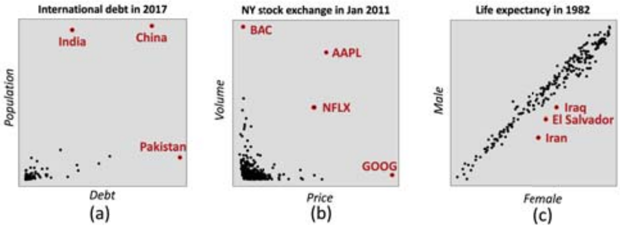


Figure 2: Examples of bivariate outliers which might not be detectable in the marginal distributions: (a) International debt, (b) New York stock exchange, and (c) Life expectancy of countries retrieved from the World Bank Database.

3.2 Design choices

3.2.1 Binning

Before identifying outliers, we perform aggregation on the data points in scatterplots based on the following observations:

- outliers and inliers are individual observations which are distinctly distributed on a scatterplot. Binning process allows us to focus on bins with a single item. In other words, we only apply our leave-one-out approach to our singleton bins because removing an observation from a dense bin will not affect the outlying scores. This also helps to reduce the computing time significantly. We show that in Section 4.6.

- We aggregate the points in each scatterplot into a certain number of bins, and then MST is computed on non-empty bins. Therefore, the complexity of our outlier detection algorithm is independent of the number of observation (*n*). Consequently, *Outliagnostics* scales well with large datasets.

There are two standard ways to bin scatterplots: Hexagon vs. Leader algorithm [19]. While hexagon binning produces regular grids, leader binning starts at the positions of data points and might produce partially overlapping coverages. Both algorithms cost linear time, but we select to use leader binning since hexagon binning is sensitive to Box Plot Rule as the distance between neighboring hexagons is always the same. Moreover, while the hexagons are fixed (independent to the distribution of data points), leaders are located at the center of the clusters and hence produce smaller mean square errors [18]. Figure 3 shows an example of hexagon binning vs. leader binning on the same input data on the left. The size of each leader indicates its coverage while the intensity of the ball highlights its density.

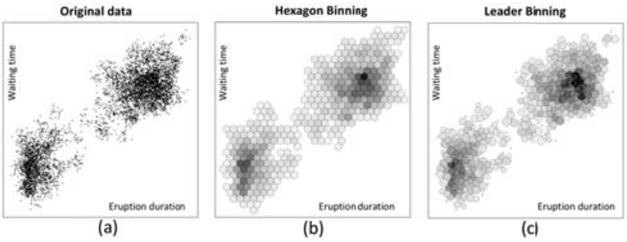


Figure 3: Old Faithful Geyser data: (a) Scatterplot of *eruption duration* vs. *waiting time* between two consecutive eruption (b) Hexagon bins (c) Leader bins.

Using the leader algorithm, we group data points in each scatterplot into a range from 50 to 250 clusters based on Euclidean distances among points. If there are more than 250 clusters, we increase the coverage radius and rebin. If there are less than 50 clusters, we reduce the coverage radius and rebin. The choice of coverage radius is constrained by efficiency (too many leaders slow down calculations of the Delaunay triangulation and MST) and sensitivity (too few and large clusters obscure features in the scatterplots and impact Box Plot Rule described in Section 2.1). Since we are focusing on detecting outliers (or observations that appears visually deviated from the others in a scatterplot [20]), increasing the number of bins to get to a reasonable representation of the scatterplot of large datasets (large *n*) does not help in highlighting outliers. In other words, we are interested in singleton and isolated bins rather than partitioning a crowded cluster into multiple smaller bins to get finer details.

3.2.2 Standardization for leave-one-out

When leave-one-out is applied, we do not re-standardize the remaining data points in the scatterplots. With a *left-out* data point, the resulting MST could be very different as depicted in Figure 4. However for many cases, the effect of removing 1 data point from the scatterplot is only on a local branch. Therefore, the MST computation time can still be improved further by localizing and recomputing only the affected branches. Moreover, we keep the same *upperbound* of the Box Plot rule for computing the outlying score of all leave-one-out plots for two reasons: (1) the Box Plot rule is sensitive to the number of observations (*n*) as it uses 3^{rd} *quartile value* and 1^{st} *quartile value* to compute the IQR. And (2) an outlier in a leave-one-out plot might not be an outlier in the original scatterplot if one of the bound is modified. Reusing the *upperbound* on MST lengths of the original plots for computing outlying scores on leave-one-out plots makes our approach more robust and faster. Figure 4 shows an example of incorrect outlying results if we do not reuse the *upperbound* of the original scatterplot for computing outlying scores of a leave-one-out plot. In particular, removing the outlier at the red arrow in Figure 4(b) results in the different MST in Figure 4(c). Notably, the new outlying score of the leave-one-out plot is even higher than the original plot (0.26 vs. 0.21) which is incorrect.

3.2.3 Parallel computing

The leave-one-out approach is computationally expensive, even with the use of binning to reduce the number of calculations needed. Fortunately, many current devices and operating systems, including mobile platforms (e.g., iPhone and Android) support parallel computing with multiple cores. So, the heavy calculation could be

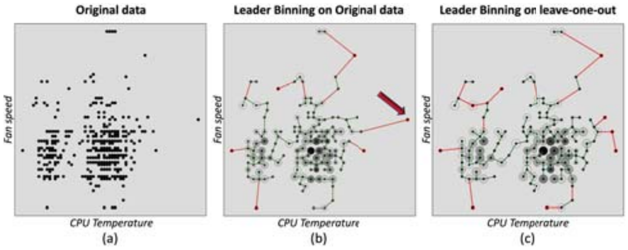


Figure 4: High Performance Computing center data: (a) Scatterplot of *CPU temperature* vs. *fan speed*. Each data point is a computer (b) Leader bins and MST of the original scatterplots (c) Leader bins and MST of the leave-one-out plot.

performed in parallel with greater efficiency. Also, doing long computation tasks in the background will not block the user interfaces and interactions of our visualization.

We developed our algorithms to support parallel computation of the outlying scores. As shown in Section 4, using concurrent calculations improves the computation time about three times or even more with higher hardware concurrency support. Benefits of parallel computation come with its cost of creation and communication overhead, as of our experiment, the number of parallel computations should be close or equal to the number of hardware concurrency support (e.g., in JavaScript it is determined as *navigator.hardwareConcurrency*).

3.3 *Outliagnostics* algorithm

Algorithm 1 describes how to compute the upper bound for the box plot rule.

Algorithm 1 Box Plot Rule to identify MST outliers

```
procedure COMPUTETHREDSHOLD(mst)
    // sort the MST by increasing order of edge lengths
    sortMSTEdgeLengths(mst)
    i50 = mst.length / 2
    i25 = i50 / 2
    i75 = i50 + i25
    return mst[i75] + 1.5 * (mst[i75] - mst[i25])
```

In Algorithm 2, we store the *upperbound* at line 5 and use it to compute the outlying score of the original scatterplot. For other leave-one-out scatterplots, we do not need to recompute *upperbound* but reuse the same *upperbound* of the original scatterplot for the new outlying computation at line 12.

Algorithm 2 Algorithm for computing outlying

```
1:  procedure COMPUTOUTLYINGSCORES(binnedData)
2:      // compute MST of the original scatterplot
3:      mst = computeMST(binnedData)
4:      // compute the upperbound of MST lengths
5:      upper = computeThredshold(mst)
6:      // compute outlying score of the original scatterplot
7:      outlying = MSTOutliers(upper,mst)
8:      for each data point d do
9:          newBinnedData = leave-one-out(d, binnedData)
10:         newMST = computeMST(newBinnedData)
11:         // Reuse the upperbound of original scatterplot
12:         newOutlying = MSTOutliers(upper,newMST)
```

3.4 *Outliagnostics* Components

This section explains our approach and its applications in detail. Our general approach is similar to other typical data visualization solutions starting from statistical quantifications to visualization overviews supported with interactions for exploration and details [34]. Figure 5 shows a schematic overview of *Outliagnostics*:

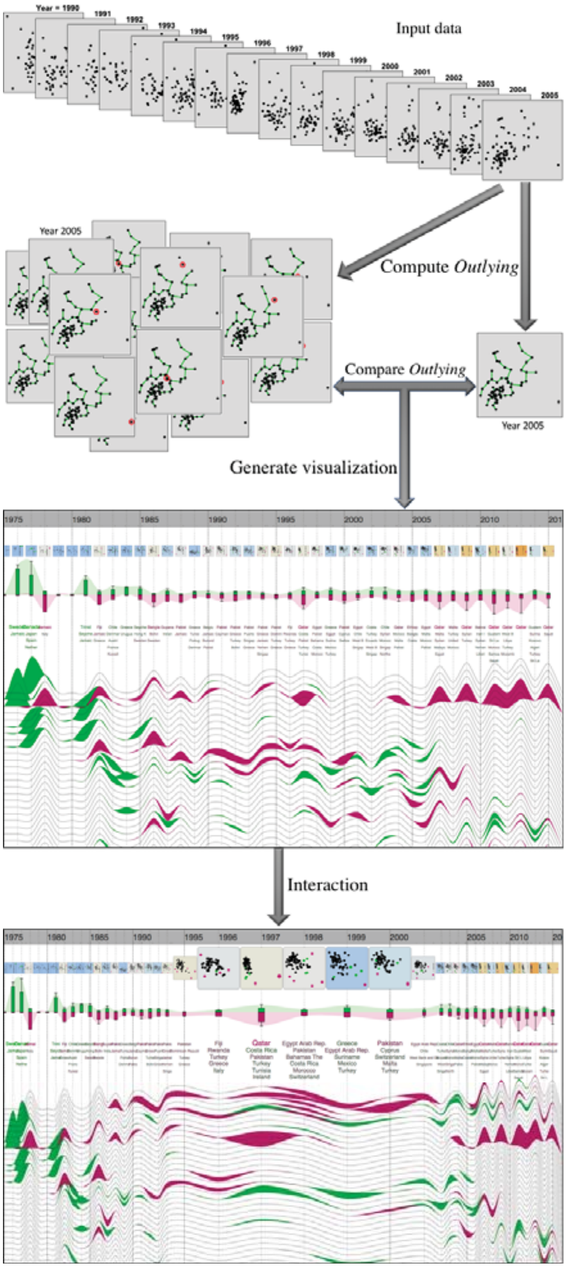


Figure 5: A schematic overview showing the main components of *Outliagnostics*: Computing outlying scores, comparing leave-one-out outlying scores to the original scatterplots, generating the visualization, and supporting interactions. The outlying time series are color-coded based on their outlying differences when leave-one-out is applied: purple for lower while green for higher than the outlying score of original scatterplot at each time point.

1. **Processing:** Our approach computes the outlying measure of each pairwise projections in the time series. Then we repeatedly leave a data point of the plot and recompute the outlying score. Differences in outlying scores between the new and original scatterplots are recorded. We use several strategies

to reduce the computation time such as binning, “selective” leave-one-out, and parallel computing as described in 3.2.

2. **Visualization:** For each variable, we display a temporal profile of outlying differences after leaving it out of the original scatterplot at each time step. We color-code observation profile. Purple is when removing the observation, the new outlying score is lower than the original scatterplot. Green is when removing the observation, the new outlying score is higher than the original scatterplot.

3. **Interaction:** The *Outliagnostics* prototype supports filtering, ordering, brushing and linking, and lensing.

Lensing area, scatterplot time series, and tag clouds: These are the top three components of the *Outliagnostics* interface as shown in Figure 1 and zoomed in for details in Figure 6. The top bar shows the timestamps where lensing is applied on mouseover, below which is the set of corresponding scatterplots: red for high outlying, blue for low outlying. This background color scheme helps users to discern the scatterplots with higher outlying scores. Below the scatterplots are the tag clouds showing the top five observations with the highest impacts (increasing or reducing) to the outlying score when leave-one-out is applied. The color of a text in these clouds also indicate if they are inliers (green) or outliers (purple).

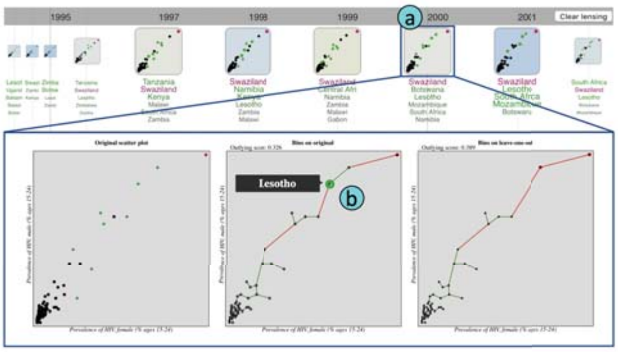


Figure 6: The lensing area, scatterplot time series, and top countries clouds for the *Prevalence Of HIV* dataset.

Users can bring up a close-up window of a scatterplot via mouse click. For example in Figure 6, users can click on the scatterplot for the year *2000* at (a) to show details at the bottom. The first box shows the distribution of the original data; the second box shows its MST (after binning), and the third box shows the new MST for outlying calculation when leaving *Lesotho* at (b) out. Red are outlying links in the MST which are longer than the *upperbound* in Algorithm 2.

The customized outlying boxplots: The customized outlying boxplot at each time step, as shown in Figure 1 (e) and Figure 7, summarizes the differences (in outlying) between the leave-one-out vs. the original plots. The zero line is the baseline for the boxplot. Above it is the green rectangle which extends from the zero line up to the average value of all the positive (inlying) differences, and the whisker at the top is the maximum of the inlying differences. On the other hand, the purple rectangle spans from the zero line down to the average of all the negative (outlying) variations, and the whisker at the bottom represents the maximum of the outlying differences. The green stream above the zero line up to the maximum inlying score and the purple one below it extending from zero line down to the maximum outlying score smooth the evolution of the inliers and outliers over time.

Figure 7 shows our customized boxplots for the *World Unemployment Rate* dataset. We can see that the maximum outlying difference of 0.08 (out of 1.0 as the maximum outlying score) occurred in the year 1966 (a) with several outliers such as Algeria and Oman and the maximum inlying difference of 0.08 locates in the year 1978 (b) with inliers like Lesotho, Swaziland, and Macedonia. The customized outlying boxplots and streams summarize outlying scores over time for further explorations using interactive features of the system.

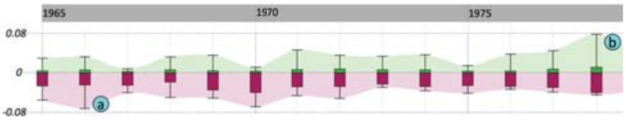


Figure 7: The customized outlying boxplots for the *World Unemployment Rate* dataset.

Instance profiles: The instance profiles allow users to dive into further details in the process of exploring outlying scores of the scatterplots overtime at the individual instance level. As depicted at the bottom panel of Figure 1 (f), this visual area contains set of outlying time series organized in descending order based on outlying/inlying scores, depending on the user preferences by selecting the options from the control panel as in Figure 1(a).

Figure 8 shows a close-up view of three countries with high impacts on the overall outlying scores overtime of the *Prevalence of HIV* dataset: Swaziland, Zimbabwe, and Lesotho. Each outlying time series is projected on a baseline (the black line at the blue arrow). The dashed curve (at the red arrow) represents the outlying scores of the original scatterplots over time. The green/purple streams above/below this dashed curve represent the increments/decrements of the outlying scores when leaving the current data item out. The type of time series representation allows the users to perceive the movements of the original outlying scores quickly as well as identify the hot spots over the long time series. In particular, each row represents a unique outlying signature of the associated entry. Notice that we can filter only outlying temporal signatures with outlying/inlying scores higher than some specific threshold using a slider. Also, the profile series are automatically ordered using their overall outlying/inlying scores (when lensing is not applied) or their outlying/inlying scores at a specific time step (when lensing is applied).

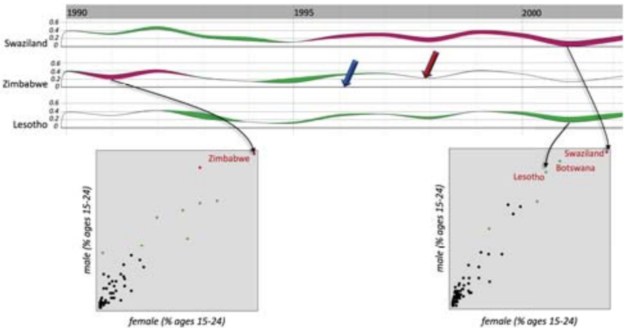


Figure 8: The profile series of three countries with high contributions to the outlying scores over years in the *Prevalence of HIV* dataset.

This section clarifies instance profile design via a use case, as shown in Figure 8. Swaziland, Zimbabwe, and Lesotho are the three countries with high impacts on the outlying score in the *Prevalence of HIV* dataset. Specifically, in 1991, Zimbabwe (at the thicker purple stream) has a high outlying score impact due to its extreme values in the prevalence of HIV for both female and male (as high as 16.6% and 6.6%). Therefore, leaving it out will reduce the outlying score of the overall scatter plot significantly (from 0.3 down to 0.19).

In 2001, Swaziland had a high impact on the outlying score due to its high values in the prevalence of HIV (30.2% for female and 8.5% for male). Leaving it out from this period will bring the outlying score of the profile from 0.13 down to 0. The outlying score of zero is suspicious, due to the reason that, in 2001, Botswana and Lesotho were also the two countries with high prevalences of HIV as 21.1%, 8.0% and 18.4%, 7.4% for female and male in these two countries correspondingly. Leaving Lesotho out of this profile makes Botswana become outlier (as it should be) in this time step and the outlying score increases up to 0.27. To address this, *Outliagnostics* represents Lesotho with a thicker green stream (higher inlying score) in this period, and also brings it on top of the instance profile when lensing is applied. The inlying score, in this case, is helpful in the sense that it shows the potential outlying score (0.26 instead of 0.13) of the original scatter plot. This inlying score acts as a warning to the users about the masking effect that Lesotho has on Botswana outlying status.

4 Experiments

4.1 Datasets

We will illustrate the features and performance of *Outliagnostics* mainly through application to various datasets. Table 1 summarizes prominent aspects of these datasets (ordered by the number of observations). The table also contains a column called *Singleton bins*. These are the average actual number of times that we have to recompute MST and use Boxplot rule for detecting outliers for each dataset. As depicted, this number is independent of *n*, but depend on the shape of data distribution shown in the last row of Table 1.

The first two datasets are from the Bureau of Labor Statistics (BLS) [9]. The US Unemployment Rate dataset contains the men and women unemployment rate of 51 States in 19 years from 1999 to 2017. The US Employment Net Change dataset contains all employees (in thousands, Month Net Change, seasonally adjusted) in Good Producing and Service Providing industries of 53 States from January 2000 to August 2018.

The next two datasets and the last one were retrieved from Kaggle [28]. The International Debt dataset from World Bank Open Data Repository [39] contains information for 124 countries from 1970 until 2024 (with projections), in particular, the debt and the population features of this dataset are used in our application. World Terrorism dataset is from National Consortium for the Study of Terrorism and Responses to Terrorism (START) [31] contains terrorism data for 205 countries over 48 years (from 1970 to 2017), and we make use of the number attacks versus the number of killed variables in our use-case. The New York Stock Exchange dataset contains the information of 501 listed Stocks from January 2010 until December 2016, the price and volume dimensions of this dataset are used in our application.

The next datasets are the World Bank Open Data Repository retrieved from UIC repository [4]. The Prevalence Of HIV dataset contains information about the prevalence of HIV female (ages 15-24) and male (in the same age range). The World Unemployment Rate dataset contains records about female and male percentage of the labor force for 241 countries over 56 years. The World Life Expectancy dataset contains male and female life expectancy of 263 countries all over the world in 56 years. To add to the variety of our datasets, we also collect health status from a High-Performance Computing Center at a university. This dataset contains fan speed and CPU temperature measurements of 467 CPUs at 33 time-steps.

4.2 Use case 1: The Life Expectancy Dataset

As depicted in Figure 9, when lensing over the 1980s period (a), the instance profile section automatically brings Iraq, El Salvador, and Iran on top of the list (b) as these three countries had higher contributions to the overall outlying scores in this period. Clicking on a time series scatterplot in 1983 to bring the details scatterplot box (c) then mouse over Iraq, Iran, and El Salvador, *Outliagnostics* shows their corresponding life expectancy for males vs. females as 66 vs. 54, 62 vs. 47, and 63 vs. 52 correspondingly. For the Iran and Iraq case, this was due to their armed conflict from 1980 to 1988; for El Salvador, it was the aftermath of its 1980-1992 Civil War.

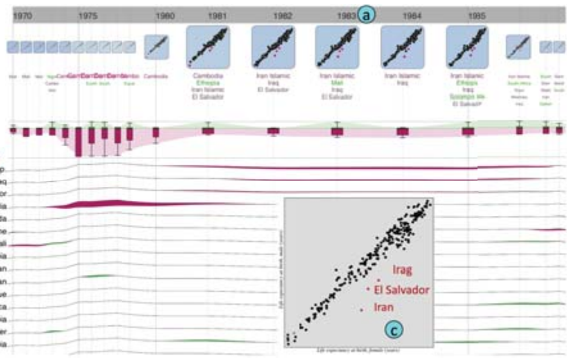


Figure 9: Lensing over the 1980s, *Outliagnostics* brings Iran, Iraq, and El Salvador on top of the item profiles section as highest contributors to the overall outlying scores in this period.

This use case shows the usefulness of each component of *Outliagnostics* visualization in identifying outliers in a time series data. At the overview level, users can quickly spot out where the “hot spots” are by viewing the fluctuation in time series boxplots to find out the places with thicker outlying/inlying streams (higher overall/potential outlying scores) to explore further. Users could further use interactive features to examine the data. For instance, users could mouse over the period with higher outlying scores in the lensing area, then clicking on the plots in the scatterplots time series section to show the *Outliagnostics* calculation details and mouse-over a data item to show its actual information at the individual data point level. Additionally, the item profile section has ordering strategies (selectable from the control panel) to bring the data items with higher contributions to the overall outlying scores to the top of the list to support further exploration.

4.3 Use case 2: US Goods and Service Employment

This use case contains US Employment Net Change data. As depicted in Figure 10, the visual interface highlights Florida as a dominant outlier in September and October 2017 and automatically brings its outlying signature to the top when users mouse-over this period in the lensing area. These sudden net changes in employment numbers are explainable due to the impacts of the hurricane Irma in September 2017, so employment number in this city was very low both in goods-producing and service-providing in September (the change were -36,900 and -130,000 employees in goods-producing correspondingly) and got back sharply in October 2017 (+37,500 and +142,000) when the hurricane had gone.

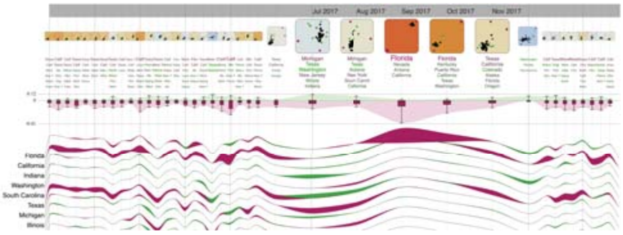
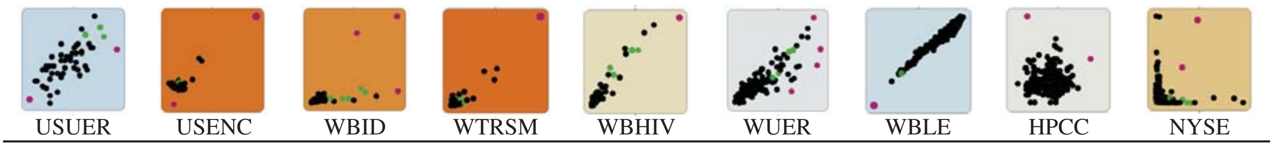


Figure 10: Florida as a dominant outlier in September 2017.

Table 1: Prominent attributes of datasets and their example scatterplots (at the bottom) used to demonstrate our *Outliagnostics*.

No.	Abbreviation	Dataset	Variable 1	Variable 2	Time steps	Instances (n)	Singleton bins
1	USUER	US Unemployment Rate	Men	Women	19	51	48
2	USENC	US Employment Net Change	Goods	Service	224	53	48
3	WBID	International Debt Data	Total debt	Population	55	124	35
4	WTRSM	World Terrorism	Attacks	Killed	48	205	39
5	WBHIV	Prevalence Of HIV	Female	Male	56	217	41
6	WUER	World Unemployment Rate	Female	Male	56	241	47
7	WBLE	World Life Expectancy	Female	Male	56	263	27
8	HPCC	High-Performance Computing	CPU Temperature	Fan Speed	33	467	63
9	NYSE	New York Stock Exchange	Price	Volume	84	501	21



4.4 Use case 3: High-Performance Computing Center

In this use case, we use *Outliagnostics* to monitor the health status of a High-Performance Computing Center (HPCC) at a university. In particular, the two variables being monitored are *CPU temperature* and *fan speed*.

Figure 11 shows an event at 12 PM on Wednesday, September 26, 2018: the CPU on *compute-3-13* suddenly became overheated. *Outliagnostics* was able to pick up the event (a) and alerted system administrator to make CPU replacement for the malfunction CPU before it harms other neighboring CPUs. As shown in the scatterplot (b), fan speeds on *compute-3-12*, *compute-3-11*, and *compute-3-10* had also pumped their fan rates as they sensed the heat from *compute-3-13*. We discussed this thermal excursion through an informal interview with Dell experts and the HPCC director. They value the diagnostics from our prototype and suggested thermal experts and hardware team to investigate this interesting correlation between CPU temperature and fan speed. The experts commented that "visual analytics provide an excellent opportunity to explore the correlation of hardware features" or "understanding the relationship of different health services is essential in our hardware design process". *Outliagnostics* is currently deploying additional dimensions, such as real-time memory usage, power consumption, CPU load, I/O bandwidth, among other integration in this on-going collaboration.

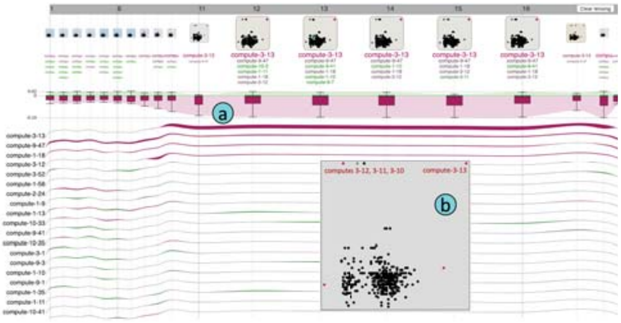


Figure 11: Monitoring health status of HPCC on Wednesday, September 26, 2018: *Outliagnostics* highlights *compute-3-13* experiencing overheat problem and its affect on the nearby CPUs.

4.5 Implementation

Outliagnostics is implemented in JavaScript using the D3.js library [7]. Our outlying computation and leave-one-out approach are also provided in form of JavaScript libraries. The online *Outliagnostics* prototype, demo video, source code, and more examples are available on our Github repository at `https://outliagnostics.github.io/`.

4.6 Evaluation on Running Times

4.6.1 Per scatterplot computation break-down

In this section, we focus on evaluating the running times of *Outliagnostics* prototype on datasets of various sizes where n is the number of instances (data points in each scatterplot). All tests were performed on a computer with 2.9 GHz Intel Core i5, macOS Sierra Version 10.12.1, 8 GB RAM. Figure 12 shows computation times broken down into the time to bin the n data points, to compute Delaunay triangulation, to compute MST, and to calculate the outlying score using the Box Plot rule. In this figure, datasets are listed from left to right in the increasing order of n (same order as in the Table 1). Here are some observations from empirical analysis:

- In Figure 12(a), WTRSM takes the most time to bin since the data distribution in a scatterplot is sparse as shown in Table 1. In other words, it takes a lot of time to find the right coverage radius for at least 50 leaders. In contrast, data points in USUER and WUER are well spread and take less time to come up with the number of clusters within the range from 50 to 250. Binning is done once per scatterplot (the same binning result are reused for all leave-one-out plots). Figure 12(b) and (c) focus on time to compute triangulation (orange), to compute MST (green), and to calculate the outlying score (red).
- Figure 12(b) shows outlying computation time of original scatterplots, which is averaged over the entire time series. As depicted, computing MST (green) is the most expensive step while calculating the outlying score using the Box Plot rule is fast. HPPC requires the most time since most of its scatterplots are dense and hence require more time to form the MST.
- Figure 12(c) shows the average outlying computation time for leave-one-out plots. As the number of data points in scatterplots increases (going from left to right), the total computation time decreases. This is because our outlying computation time is independent of n (but dependent on how the data distribution looks like: sparse or dense). Most importantly, as discussed in section 3, our algorithm checks if the *left-out* data point is in a singleton cluster or not. If it is, we perform the three steps in computing the outlying score: computing triangulation, forming MST, and applying the Box Plot rule. Otherwise, we skip since removing a data point from a larger cluster will not affect the final outlying score.

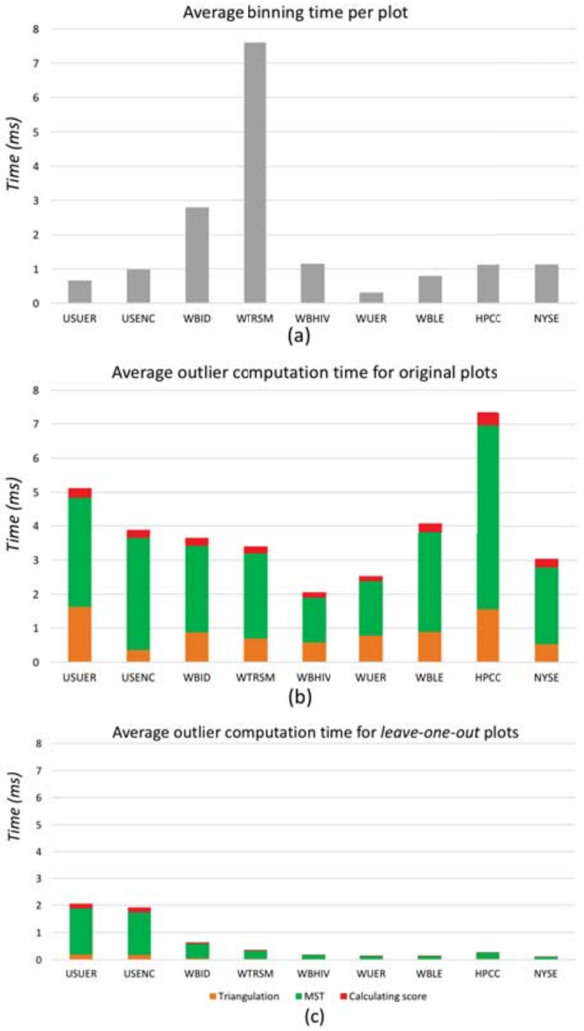


Figure 12: Computation times (in milliseconds) for datasets with various sizes where n is the number of instances. Datasets are listed from left to right in the increasing order of n.

Overall, our proposed approach scales well with larger datasets containing thousands of observations. In other words, our leave-one-out strategy does not depend on n, but depends on the number of single data points (no other data point in their proximal surrounding). In fact, these are the data points that might be able to create masking and swamping effects.

4.6.2 Running times for different parallelizations

In this section, we evaluate the running times of *Outliagnostics* prototype with different settings to find the best parallelism configuration(s). The total of 23,714 (original and leave-one-out scatterplots) from the datasets described in section 4.1 were tested. These datasets were executed 30 times in each setting (to make sure the reported execution times are stable and not happened solely by chances due to the stochastic nature of computer execution time) then the averaged execution times are reported as in Figure 13. All tests were performed on two computers, the first computer with 2.9 GHz Intel Core i5, macOS Mojave Version 10.14.3, 8 GB RAM, and hardware concurrency = 4 and the second one with 2.6 GHz Intel Core i7, macOS Mojave Version 10.14.3, 16 GB RAM, and hardware concurrency = 12. The parallelism configurations are 1, 4, 8, 12, and 16 web workers correspondingly. We explored these number of workers because they are around the number of hardware concurrency of the testing platforms (4, and 12 respectively).

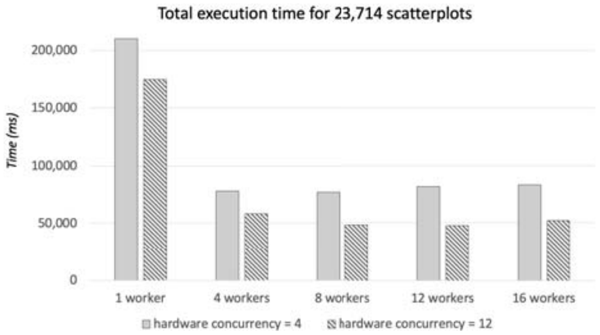


Figure 13: Computation times (in milliseconds) for 23,714 (original and leave-one-out scatterplots) from datasets with various sizes in different parallelism configurations.

Generally, using multiple workers helps to improve the computation time about three times and even more in devices with higher concurrency support. Also, the benefits of running more workers come with the cost of their creation and communication overheads, as of our experiment, a number of workers close or equal to the hardware concurrency support will give the best performance. The hardware concurrency support for different devices and operating systems could be determined in JavaScript as *navigator.hardwareConcurrency*, and this is the default parallelization setting of our system.

4.7 Extending to higher-dimensional data

The concepts discussed in this paper are specific to two dimensional (2D) temporal datasets. However, they could be generalized to detect outliers for a higher number of dimensions (nD) time series with a few modifications. In term of outlying score computation, we need to extend the Euclidean distance calculation from 2D to nD. In the nD version, we could also explore other options for distance calculation to avoid the "curse of dimensionality". For instance, the Manhattan distance metric (L_1 norm) might be preferable than the Euclidean distance metric (L_2 norm), or even the L_k norm where k is a fraction should be explored [3]. The visualization components and interactive operations of our prototype remain valid, except that the radar-charts can be used to replace scatterplots.

5 Conclusion

In this paper, we have proposed a new approach for visualizing the outlying temporal profile of each data entry with respect to the overall distributions in two-dimensional temporal datasets and also discussed the extension of the ideas to a higher number of dimensions. Our approach is based on the leave-one-out strategy for measuring the significance of individual data points in computing outlying as a whole. This approach not only allows us to detect multivariate outliers but avoid both masking and swamping effects. We demonstrated our *Outliagnostics* prototype on various use cases of the US employment data, social and economic data from the World Bank database, and health status of high-performance computing systems. We also evaluated computing times to provide users with an idea of how long it takes to use our approach for certain datasets. The running time evaluations prove that our approach can scale well with large data thanks to binning, redundant checking before performing outlying computation on the leave-one-out plots, and the use of multiple web workers.

References

[1] E. Acuna and C. Rodriguez. A meta analysis study of outlier detection methods in classification. *Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez*, pp. 1–25, 2004.

[2] C. C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2013.

[3] A. Anand, L. Wilkinson, and D. N. Tuan. An l-infinity norm visual classifier. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 687–692, Dec 2009. doi: 10.1109/ICDM.2009.119

[4] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[5] V. Barnett and T. Lewis. *Outliers in statistical data.* John Wiley & Sons Ltd., 2nd edition ed., 1978.

[6] H. Beyer. Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. - menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, xvi, 688 s. *Biometrical Journal*, 23(4):413–414, 1981. doi: 10.1002/bimj.4710230408

[7] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2301–2309, 2011.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, vol. 29, pp. 93–104. ACM, 2000.

[9] Bureau of Labor Statistics. http://www.bls.gov/data/, September 2018.

[10] N. Cao, C. Lin, Q. Zhu, Y.-R. Lin, X. Teng, and X. Wen. Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data. *IEEE transactions on visualization and computer graphics*, 24(1):23–33, 2018.

[11] N. Cao, C. Shi, S. Lin, J. Lu, Y.-R. Lin, and C.-Y. Lin. Targetvue: Visual analysis of anomalous user behaviors in online communication systems. *IEEE transactions on visualization and computer graphics*, 22(1):280–289, 2016.

[12] C. Caroni and P. Prescott. On rohlf s method for the detection of outliers in multivariate data. *Journal of Multivariate Analysis*, 52(2):295–307, 1995.

[13] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. doi: 10.1145/1541880.1541882

[14] T. Dang and V. V. Pham. Timematrix: Visual representation for temporal pattern detection in dynamic networks, vast 2018 mini-challenge 3. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 108–109, Oct 2018. doi: 10.1109/VAST.2018.8802457

[15] N. I. George, J. F. Bowyer, N. M. Crabtree, and C.-W. Chang. An iterative leave-one-out approach to outlier detection in rna-seq data. *PLoS One*, 10(6):e0125224, 2015.

[16] M. Greenacre and H. Ö. Ayhan. Identifying inliers. 2014.

[17] A. S. Hadi and J. S. Simonoff. Procedures for the identification of multiple outliers in linear models. *Journal of the American Statistical Association*, 88(424):1264–1272, 1993.

[18] P. Hall and M. Wand. On the accuracy of binned kernel density estimators. *J. Multivar. Anal.*, 56(2):165–184, Feb. 1996. doi: 10.1006/jmva.1996.0009

[19] J. A. Hartigan. *Clustering algorithms*. Wiley, 1975.

[20] D. Hawkins. *Identification of outliers*. Monographs on applied probability and statistics. Chapman and Hall, London [u.a.], 1980.

[21] R. W. Hayden. A dataset that is 44% outliers. *J Stat Educ*, 13(1), 2005.

[22] B. Iglewicz and . Hoaglin, David C. (David Caster). *How to detect and handle outliers*. Milwaukee, Wis. : ASQC Quality Press, 1993. Includes bibliographical references (p. 73-78) and index.

[23] B. Iglewicz and J. Martinez. Outlier detection using robust measures of scale. *Journal of Statistical Computation and Simulation*, 15(4):285–293, 1982. doi: 10.1080/00949658208810595

[24] B. Irad. Outlier detection, data mining, and knowledge discovery handbook: A complete guide for practitioners and researchers, 2005.

[25] M. F. Jaing, S. S. Tseng, and C. M. Su. Two-phase clustering process for outliers detection. *Pattern Recogn. Lett.*, 22(6-7):691–700, May 2001. doi: 10.1016/S0167-8655(00)00131-8

[26] M. Jiang, S. Tseng, and C. Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6):691 – 700, 2001. doi: 10.1016/S0167-8655(00)00131-8

[27] J. M. Jobe and M. Pokojovy. A cluster-based outlier detection scheme for multivariate data. *Journal of the American Statistical Association*, 110(512):1543–1551, 2015. doi: 10.1080/01621459.2014.983231

[28] Kaggle. https://www.kaggle.com/datasets, September 2018.

[29] T. Kutsuna and A. Yamamoto. Outlier detection based on leave-one-out density using binary decision diagrams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 486–497. Springer, 2014.

[30] J. Lin, D. Ye, C. Chen, and M. Gao. *Minimum Spanning Tree Based Spatial Outlier Mining and Its Applications*, pp. 508–515. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-79721-0_69

[31] National Consortium for the Study of Terrorism and Responses to Terrorism (START). http://www.start.umd.edu/, September 2018.

[32] R. Pamula, J. K. Deka, and S. Nandi. An outlier detection method based on clustering. In *2011 Second International Conference on Emerging Applications of Information Technology*, pp. 253–256, Feb 2011. doi: 10.1109/EAIT.2011.25

[33] V. Pham and T. Dang. Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1296–1301, Dec 2018. doi: 10.1109/BigData.2018.8622092

[34] V. Pham and T. Dang. SOAViz: Visualization for Portable X-ray Fluorescence Soil Profiles. In R. Bujack, K. Feige, K. Rink, and D. Zeckzer, eds., *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2019. doi: 10.2312/envirvis.20191102

[35] V. V. Pham and T. Dang. Mtdes: Multi-dimensional temporal data exploration system; strong support for exploratory analysis award in vast 2018, mini-challenge 2. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 100–101, Oct 2018. doi: 10.1109/VAST.2018.8802440

[36] F. J. Rohlf. Generalization of the gap test for the detection of multivariate outliers. *Biometrics*, 31(1):93–101, 1975.

[37] L. Wilkinson. Visualizing big data outliers through distributed aggregation. *IEEE transactions on visualization and computer graphics*, 2017.

[38] L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *Proceedings of the IEEE Information Visualization 2005*, pp. 157–164. IEEE Computer Society Press, 2005.

[39] World Bank Open Data. https://data.worldbank.org/, September 2018.

[40] K. Xu, M. Xia, X. Mu, Y. Wang, and N. Cao. Ensemblelens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE transactions on visualization and computer graphics*, 25(1):109–119, 2019.

[41] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, Jan. 1971. doi: 10.1109/T-C.1971.223083

[42] T. Zhang, X. Wang, Z. Li, F. Guo, Y. Ma, and W. Chen. A survey of network anomaly visualization. *Science China Information Sciences*, 60(12):121101, 2017.