I've Seen "Enough": Incrementally Improving Visualizations to Support Rapid Decision Making

Sajjadur Rahman[1] Maryam Aliakbarpour[2] Ha Kyung Kong[1] Eric Blais[3] Karrie Karahalios[1,4]
Aditya Parameswaran[1] Ronitt Rubinfield[2]
[1]University of Illinois (UIUC) [4]Adobe Research [2]MIT [3]University of Waterloo
{srahman7,hkong6,kkarahal,adityagp}@illinois.edu {maryama@,ronitt@csail}.mit.edu eblais@uwaterloo.ca

ABSTRACT

Data visualization is an effective mechanism for identifying trends, insights, and anomalies in data. On large datasets, however, generating visualizations can take a long time, *delaying the extraction of insights, hampering decision making, and reducing exploration time*. One solution is to use online sampling-based schemes to generate visualizations faster while improving the displayed estimates incrementally, eventually converging to the exact visualization computed on the entire data. However, the intermediate visualizations are approximate, and often fluctuate drastically, leading to potentially incorrect decisions. We propose sampling-based incremental visualization algorithms that reveal the "salient" features of the visualization quickly—with a $46\times$ speedup relative to baselines—while minimizing error, thus enabling rapid and error-free decision making. We demonstrate that these algorithms are *optimal* in terms of sample complexity, in that given the level of interactivity, they generate approximations that take as few samples as possible. We have developed the algorithms in the context of an incremental visualization tool, titled INCVISAGE, for trendline and heatmap visualizations. We evaluate the usability of INCVISAGE via user studies and demonstrate that users are able to make effective decisions with incrementally improving visualizations, especially compared to vanilla online-sampling based schemes.

1. INTRODUCTION

Visualization is emerging as the most common mechanism for exploring and extracting value from datasets by novice and expert analysts alike. This is evidenced by the huge popularity of data visualization tools such as Microsoft's PowerBI and Tableau, both of which have 100s of millions of dollars in revenue just this year [4, 2]. And yet data visualization on increasingly large datasets, remains *cumbersome*: when datasets are large, generating visualizations can take hours, *impeding interaction, preventing exploration, and delaying the extraction of insights* [38]. One approach to generating visualizations faster is to sample a small number of datapoints from the dataset online; by using sampling, we can view visualizations that incrementally improve over time and eventually converge to the visualization computed on the entire data. However, such intermediate visualizations are approximate, and often fluctuate drastically, leading to *incorrect insights and conclusions*. The key question we wish to address in this paper is the following: ***can we develop a sampling-based incremental visualization algorithm that reveals the features of the eventual visualization quickly, but does so in a manner that is guaranteed to be correct?***

Illustrative Example. We describe the goals of our sampling algorithms via a pictorial example. In Figure 1, we depict, in the first row, the variation of present sampling algorithms as time progresses and more samples are taken: at t_1, t_2, t_4, t_7, and when all of the data has been sampled. This is, for example, what visualizing the results of an online-aggregation-like [20] sampling algorithm might provide. If a user sees the visualization at any of the intermediate time points, they may make incorrect decisions. For example, at time t_1, the user may reach an incorrect conclusion that the values at the start and the end are lower than most of the trend, while in fact, the opposite is true—this anomaly is due to the skewed samples that were drawn to reach t_1. The visualization continues to vary at t_2, t_4, and t_7, with values fluctuating randomly based on the samples that were drawn. Indeed, a user may end up having to wait until the values stabilize, and even then may not be able to fully trust the results. One approach to ameliorate this issue would be to use confidence intervals to guide users in deciding when to draw conclusions—however, prior work has demonstrated that users are not able to interpret confidence intervals correctly [15]. At the same time, the users are subject to the same vagaries of the samples that were drawn.

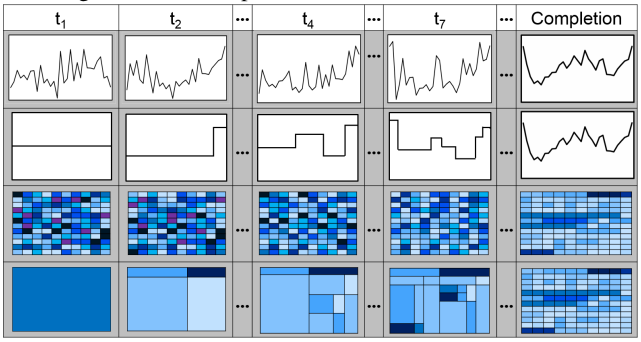


Figure 1: INCVISAGE example.

Another approach, titled INCVISAGE[1], that we espouse in this paper and depict in the second row is the following: at each time point t_i, reveal one additional segment for a i-segment trendline, by splitting one of the segments for the trendline at t_{i-1}, when the tool is confident enough to do so. Thus, INCVISAGE is very conservative at t_1 and just provides a mean value for the entire range, then at t_2, it splits the single segment into two segments, indicating that the trend increases towards the end. Overall, by t_7, the tool has indicated many of the important features of the trend: it starts off high, has a bump in the middle, and then increases towards the end. This approach reveals features of the eventual visualization in the order of prominence, allowing users to gain early insights and draw conclusions early. This approach is reminiscent of interlaced pixel-based image generation in browsers [40], where the image slowly goes from being blurry to sharp over the course of the rendering, displaying the most salient features of the visualization before the less important features. Similar ideas have also been developed in other domains such as signal processing [50] and geo maps [13].

So far, we've described trendlines; the INCVISAGE approach can be applied to heatmap visualizations as well—depicted in row 4 for

[1]INCVISAGE is a portmanteau of "Inc", i.e., short for incremental, and "Envisage", i.e., to visualize.

the corresponding online-aggregation-like approach shown in row 3—as is typical in heatmaps, the higher the value, the darker the color. Here, there is no notion of confidence intervals, so row 3 is our current best approach for depicting the approximate heatmap. Once again row 3—the status quo—fluctuates tremendously, not letting analysts draw meaningful insights early and confidently. On the other hand, row 4—the INCVISAGE approach—repeatedly subdivides a block into four blocks when it is confident enough to do so, emphasizing early, that the right hand top corner has a higher value, while the values right below it are somewhat lower. In fact, the intermediate visualizations may be preferable because users can get the big picture view without being influenced by noise.

Open Questions. Naturally, developing INCVISAGE brings a whole host of open questions, that span the spectrum from usability to algorithmic process. First, it is not clear at what rate we should be displaying the results of the incremental visualization algorithm. When can we be sure that we know enough to show the ith increment, given that the $(i-1)$th increment has been shown already? How should the ith increment differ from the $(i-1)$th increment? How do we prioritize sampling to ensure that we get to the ith increment as soon as possible, but with guarantees? Can we show that our algorithm is in a sense 'optimal', in that it aims to take as few samples as possible to display the ith increment with guarantees? And at the same time, how do we ensure that our algorithm is lightweight enough that computation doesn't become a bottleneck? How do we place the control in the user's hands in order to control the level of interactivity needed?

The other open questions involved are related to how users interpret incremental visualizations. Can users understand and make sense of the guarantees provided? Can they use these guarantees to make well-informed decisions and terminate early without waiting for the entire visualization to be generated?

Contributions. In this paper, we address all of these open questions. Our primary contribution in the paper is the notion of *incrementally improving visualizations that surface important features as they are determined with high confidence* — bringing a concept that is commonly used in other settings, e.g., rasterization and signal processing, to visualizations. Given a user specified interactivity threshold (described later), we develop incremental visualizations algorithms for INCVISAGE that minimizes error. We introduce the concept of *improvement potential* to help us pick the right improvement per increment. We find, somewhat surprisingly, that a remarkably simple algorithm works best under a *sub-Gaussian assumption* [45] about the data, which is reasonable to assume in real-world datasets (as we show in this paper). We further demonstrate that these algorithms are *optimal* in that they generate approximations within some error bound given the interactivity threshold. When users don't provide their desired interactivity threshold, we can pick appropriate parameters that help them best navigate the tradeoff between error and latency. We additionally perform user studies to evaluate the usability of an incremental visualization interface, and evaluate whether users are able to make effective decisions with incrementally improving visualizations. We found that they are able to effectively determine when to stop the visualization and make a decision, trading off latency and error, especially when compared to traditional online sampling schemes.

Prior Work. Our work is complementary to other work in the incremental visualization space. In particular, sampleAction [17] and online aggregation [20] both perform online sampling to depict aggregate values, along with confidence-interval style estimates to depict the uncertainty in the current aggregates. Online aggregation presents these values as raw values, while sampleAction displays the corresponding bar chart. In both cases, however, these approaches prevent users from getting early insights since they need to wait for the values to stabilize. As we will discuss later, our approach can be used in tandem with online aggregation-based approaches. IFOCUS [32], PFunk-H [10], and ExploreSample [53] are other approximate visualization algorithms targeted at generating visualizations rapidly while preserving perceptual insights. IFOCUS emphasizes the preservation of pairwise ordering of bars in a bar chart, as opposed to the actual values; PFunk-H uses perceptual functions from graphical perception research to terminate visualization generation early; ExploreSample approximates scatterplots, ensuring that overall distributions and outliers are preserved. Lastly, M4 [29] uses rasterization to reduce the dimensionality of a time series without impacting the resulting visualization. None of these methods emphasize revealing features of visualizations incrementally.

Outline. The outline of the remainder of this paper is as follows: in Section 2 we formally define the incremental visualization problem. Section 3 outlines our incremental visualization algorithm while Section 4 details the system architecture of INCVISAGE. In Section 5 we summarize the experimental results and the key takeaways. Then we present the user study design and outcomes in Section 6 (for usability) and 7 (for comparison to traditional online sampling schemes). Section 8 describes other related work on data visualization and analytics.

2. PROBLEM FORMULATION

In this section, we first describe the data model, and the visualization types we focus on. We then formally define the problem.

2.1 Visualizations and Queries

Data and Query Setting. We operate on a dataset consisting of a single large relational table R. Our approach also generalizes to multiple tables obeying a star or a snowflake schemata but we focus on the single table case for ease of presentation. As in a traditional OLAP setting, we assume that there are dimension attributes and measure attributes—dimension attributes are typically used as group-by attributes in aggregate queries, while measure attributes are the ones typically being aggregated. For example, in a product sales scenario, day of the year would be a typical dimension attribute, while the sales would be a typical measure attribute.

INCVISAGE supports two kinds of visualizations: trendlines and heatmaps. These two types of visualizations can be translated to aggregate queries Q_T and Q_H respectively:

```
Q_T = SELECT X_a, AVG(Y) FROM R
      GROUP BY X_a ORDER BY X_a,   and
Q_H = SELECT X_a, X_b, AVG(Y) FROM R
      GROUP BY X_a, X_b ORDER BY X_a, X_b
```

Here, X_a and X_b are dimension attributes in R, while Y is a measure attribute being aggregated. The trendline and heatmap visualizations are depicted in Figure 1. For trendlines (query Q_T), the attribute X_a is depicted along the x-axis while the aggregate AVG(Y) is depicted along the y-axis. On the other hand, for heatmaps (query Q_H) the two attributes X_a and X_b are depicted along the x-axis and y-axis, respectively. The aggregate AVG(Y) is depicted by the color intensity for each block (i.e., each X_a, X_b combination) of the heatmap. Give a continuous color scale, the higher the value of AVG(Y), the higher the color intensity. The complete set of queries (including WHERE clauses and other aggregates) that are supported by INCVISAGE can be found in Section 3.5—we focus on the simple setting for now.

Note that we are implicitly focusing on X_a and X_b that are *ordinal*, i.e., have an order associated with them so that it makes sense to visualize them as a trendline or a heatmap. As we will demonstrate subsequently, this order is crucial in letting us approximate portions of the visualization that share similar behavior. While our techniques could also be applied to X_a, X_b that are not ordinal by

enforcing some order, e.g., a lexicographic order, the resulting visualizations are not as meaningful.

Sampling Engine. We assume that we have a sampling engine that can efficiently retrieve random tuples from R corresponding to different values of the dimension attribute(s) X_a and/or X_b (along with optional predicates from a WHERE). Focusing on Q_T for now, given a certain value of $X_a = a_i$, our engine provides us a random tuple that satisfies $X_a = a_i$. Then, by looking up the value of Y corresponding to this tuple, we can get an estimate for the average of Y for $X_a = a_i$. Our sampling engine is drawn from Kim et al. [32] and maintains an in-memory bitmap on the dimension attributes, allowing us to quickly identify tuples matching arbitrary conditions [33]. Bitmaps are highly compressible and effective for read-only workloads [34, 52, 51], and have been applied recently to sampling for approximate generation of bar charts [32].

Thus, given our sampling engine, we can retrieve samples of Y given $X_a = a_i$ (or $X_a = a_i \wedge X_b = b_i$ for heat maps). We call the multiset of values of Y corresponding to $X_a = a_i$ across all tuples to be a *group*. This allows us to say that we are *sampling from a group*, where implicitly we mean we are sampling the corresponding tuples and retrieving the Y value.

Next, we describe our problem of incrementally generating visualizations more formally. We focus on trendlines (i.e., Q_T); the corresponding definitions and techniques for heatmaps are described in Appendix A.

2.2 Incremental Visualizations

From this point forward, we describe the concepts in the context of our visualizations in row 2 of Figure 1.

Segments and k-Segment Approximations. We denote the cardinality of our group-by dimension attribute X_a as m, i.e., $|X_a| = m$. In Figure 1 this value is 36. At all time points over the course of visualization generation, we display one value of AVG(Y) corresponding to each group $x_i \in X_a, i \in 1 \ldots m$—thus, the user is always shown a complete trendline visualization. To approximate our trendlines, we use the notion of *segments* that encompass multiple groups. We define a segment as follows:

Definition 1. *A* segment S *corresponds to a pair* (I, η)*, where* η *is a value, while* I *is an interval* $I \subseteq [1, m]$ *spanning a consecutive sequence of groups* $x_i \in X_a$*.*

For example, the segment S $([2, 4], 0.7)$ corresponds to the interval of x_i corresponding to x_2, x_3, x_4, and has a value of 0.7. Then, a k-segment approximation of a visualization comprises k disjoint segments that span the entire range of $x_i, i = 1 \ldots m$. Formally:

Definition 2. *A* k-segment *approximation is a tuple* $L_k = (S_1, \ldots, S_k)$ *such that the segments* $S_1, \ldots, S_k$ *partition the interval* $[1, m]$ *into* k *disjoint sub-intervals.*

In Figure 1, at t_2, we display a 2-segment approximation, with segments $([1, 30], 0.4)$ and $([31, 36], 0.8)$; and at t_7, we display a 7-segment approximation, comprising $([1, 3], 0.8)$, $([4, 14], 0.4)$, …, and $([35, 36], 0.7)$. When the number of segments is unspecified, we simply refer to this as a *segment* approximation.

Incrementally Improving Visualizations. We are now ready to describe our notion of incrementally improving visualizations.

Definition 3. *An* incrementally improving visualization *is defined to be a sequence of* m *segment approximations,* $(L_1, \ldots, L_m)$*, where the* i*th item* $L_i, i > 1$ *in the sequence is a* i*-segment approximation, formed by selecting one of the segments in the* $(i-1)$*-segment approximation* L_{i-1}*, and dividing that segment into two.*

Thus, the segment approximations that comprise an incrementally improving visualization have a very special relationship to each other: each one is a *refinement* of the previous, revealing one new *feature* of the visualization and is formed by dividing one of the segments S in the i-segment approximation into two new segments to give an $(i+1)$-segment approximation: we call this process *splitting a segment*. The group within $S \in L_i$ immediately following which the split occurs is referred to as a *split group*. Any group in the interval $I \in S$ except for the last group can be chosen as a *split group*. As an example, in Figure 1, the entire second row corresponds to an incrementally improving visualization, where the 2-segment approximation is generated by taking the segment in the 1-segment approximation corresponding to $([1, 36], 0.5)$, and splitting it at group 30 to give $([1, 30], 0.4)$ and $([31, 36], 0.8)$. Therefore, the *split group* is 30. The reason why we enforce two neighboring segment approximations to be related in this way is to ensure that there is *continuity* in the way the visualization is generated, making it a smooth user experience. If, on the other hand, each subsequent segment approximation had no relationship to the previous one, it could be a very jarring experience for the user with the visualizations varying drastically, making it hard for them to be confident in their decision making. We show in Section 5 that removing this restriction results in visualizations that are not significantly better in terms of error, but are much more costly to compute.

Underlying Data Model and Output Model. To characterize the performance of an incrementally improving visualization, we need a model for the underlying data. We represent the underlying data as a sequence of m distributions $D_1, \ldots, D_m$ with means $\mu_1, \ldots, \mu_m$ where, $\mu_i = \text{AVG(Y)}$ for $x_i \in X_a$. To generate our incrementally improving visualization and its constituent segment approximations, we draw samples from distributions $D_1, \ldots, D_m$. Our goal is to approximate the mean values $(\mu_1, \ldots, \mu_m)$ while taking as few samples as possible.

The output of a k-segment approximation L_k can be represented alternately as a sequence of values $(\nu_1, \ldots, \nu_m)$ such that ν_i is equal to the value corresponding to the segment that encompasses x_i, i.e., $\forall_{x_i \in S_j} \nu_i = \eta_j$, where $S_j(I, \eta_j) \in L_k$. By comparing $(\nu_1, \ldots, \nu_m)$ with $(\mu_1, \ldots, \mu_m)$, we can evaluate the error corresponding to a k-segment approximation, as we describe later.

Incrementally Improving Visualization Generation Algorithm. Given our data model, an incrementally improving visualization generation algorithm proceeds in *iterations*: at the ith iteration, this algorithm takes some samples from the distributions $D_1, \ldots, D_m$, and then at the end of the iteration, it outputs the i-segment approximation L_i. Thus, a certain number of samples are taken in each iteration, and one segment approximation is output at the end of it. We denote the number of samples taken in iteration i as N_i. When L_m is output, the algorithm terminates.

2.3 Characterizing Performance

There are multiple ways to characterize the performance of incrementally improving visualization generation algorithms.

Sample Complexity, Interactivity and Wall-Clock Time. The first and most straightforward way to evaluate performance is by measuring the samples taken in each iteration k, N_k, i.e., the *sample complexity*. Since the time taken to acquire the samples is proportional to the number of samples in our sampling engine (as shown in [32]), this is a proxy for the time taken in each iteration. Recent work has identified 500ms as a "rule of thumb" for interactivity in exploratory visual data analysis [38], beyond which analysts end up getting frustrated, and as a result explore fewer hypotheses. To enforce this rule of thumb, we can ensure that our algorithms take only as many samples per iteration as is feasible within 500ms — a time budget. We also introduce a new metric called interactivity that quantifies the overall user experience:

$$\lambda = \frac{\sum_{k=1}^{m} N_k \times (m - k + 1)}{k'}$$

where N_k is the number of samples taken at iteration k and k' is

the number of iterations where $N_k > 0$. The larger the λ, the smaller the interactivity: this measure essentially captures the average waiting time across iterations where samples are taken. We explore the measure in detail in Section 3.4 and 5.5. A more direct way to evaluate performance is to measure the wall-clock time for each iteration.

Error Per Iteration. Since our incrementally improving visualization algorithms trade off taking fewer samples to return results faster, it can also end up returning segment approximations that are incorrect. We define the ℓ_2 *squared error* of a k-segment approximation L_k with output sequence $(\nu_1, \ldots, \nu_m)$ for the distributions $D_1, \ldots, D_m$ with means $\mu_1, \ldots, \mu_m$ as

$$\text{err}(L_k) = \frac{1}{m} \sum_{i=1}^{m} (\mu_i - \nu_i)^2 \tag{1}$$

We note that there are several reasons a given k-segment approximation may be erroneous with respect to the underlying mean values $(\mu_1, \ldots, \mu_m)$: (1) We are representing the data using k-segments as opposed to m-segments. (2) We use incremental refinement: each segment approximation builds on the previous, possibly erroneous estimates. (3) The estimates for each group and each segment may be biased due to sampling.

These types of error are all unavoidable — the first two reasons enable a visualization that incrementally improves over time, while the last one occurs whenever we perform sampling: (1) While a k-segment approximation does not capture the data exactly, it provides a good approximation preserving visual features, such as the overall major trends and the regions with a large value. Moreover, computing an accurate k-segment approximation requires fewer samples and therefore faster than an accurate m-segment approximation. (2) Incremental refinement allows users to have a fluid experience, without the visualization changing completely between neighboring approximations. At the same time, is not much worse in error than visualizations that change completely between approximations, as we will see later. (3) And lastly, sampling is necessary for us to return visualizations faster, but perhaps at the cost of erroneous estimates.

2.4 Problem Statement

The goal of our incrementally improving visualization generation algorithm is, at each iteration k, generate a k-segment approximation that is not just "close" to the best possible refinement at that point, but also takes the least number of samples to generate such an approximation. Further, since the decisions made by our algorithm can vary depending on the samples retrieved, we allow the user to specify a failure probability δ, which we expect to be close to zero, such that the algorithm satisfies the "closeness" guarantee with probability at least $1 - \delta$.

Problem 1. *Given a query Q_T, and the parameters δ, ϵ, design an incrementally improving visualization generation algorithm that, at each iteration k, returns a k-segment approximation while taking as few samples as possible, such that with probability $1 - \delta$, the error of the k-segment approximation L_k returned at iteration k does not exceed the error of the best k-segment approximation formed by splitting a segment of L_{k-1} by no more than ϵ.*

3. VISUALIZATION ALGORITHMS

In this section, we gradually build up our solution to Problem 1. We start with the ideal case when we know the means of all of the distributions up-front, and then work towards deriving an error guarantee for a single iteration of an incrementally improving visualization algorithm. Then, we propose our incrementally improving visualization generation algorithm *ISplit* assuming the same guarantee across all iterations. We further discuss how we can tune the guarantee across iterations in Section 3.4. We consider extensions to other query classes in Section 3.5. We can derive similar algorithms and guarantees for heatmaps in Appendix A.

3.1 Case 1: The Ideal Scenario

We first consider the ideal case where the means $\mu_1, \ldots, \mu_m$ of the distributions $D_1, \ldots, D_m$ are known. Then, our goal reduces to obtaining the best k segment approximation of the distributions at iteration k, while respecting the refinement restriction. Say the incrementally improving visualization generation algorithm has obtained a k-segment approximation L_k at the end of iteration k. Then, at iteration $(k+1)$, the task is to identify a segment $S_i \in L_k$ such that splitting S_i into two new segments T and U minimizes the error of the corresponding $(k+1)$-segment approximation L_{k+1}. We describe the approach, followed by its justification.

Approach. At each iteration, we split the segment $S_i \in L_k$ into T and U that maximizes the quantity $\frac{|T|\cdot|U|}{|S_i|\cdot m}(\mu_T - \mu_U)^2$. Intuitively, this quantity—defined below as the *improvement potential* of a refinement—picks segments that are large, and within them, splits where we get roughly equal sized T and U, with large differences between μ_T and μ_U.

Justification. The ℓ_2 *squared error* of a segment S_i (I_i, η_i), where $I_i = [p, q]$ and $1 \le p \le q \le m$, for the distributions $D_p, \ldots, D_q$ with means $\mu_p, \ldots, \mu_q$ is

$$\text{err}(S_i) = \frac{1}{(q-p+1)} \sum_{j=p}^{q} (\mu_j - \eta_i)^2 = \frac{1}{|S_i|} \sum_{j \in S_i} (\mu_j - \eta_i)^2$$

Here, $|S_i|$ is the number of groups (distributions) encompassed by segment S_i. When the means of the distributions are known, $\text{err}(S_i)$ will be minimized if we represent the value of segment S_i as the mean of the groups encompassed by S_i, i.e., setting $\eta_i = \mu_{S_i} = \sum_{j \in S_i} \mu_j / |S_i|$ minimizes $\text{err}(S_i)$. Therefore, in the ideal scenario, the error of the segment S_i is

$$\text{err}(S_i) = \frac{1}{|S_i|} \sum_{j \in S_i} (\mu_j - \eta_i)^2 = \frac{1}{|S_i|} \sum_{j \in S_i} \mu_j^2 - \mu_{S_i}^2 \tag{2}$$

Then, using Equation 1, we can express the ℓ_2 *squared error* of the k-segment approximation L_k as follows:

$$\text{err}(L_k) = \frac{1}{m} \sum_{i=1}^{m} (\mu_i - \nu_i)^2 = \sum_{i=1}^{k} \frac{|S_i|}{m} \text{err}(S_i)$$

Now, L_{k+1} is obtained by splitting a segment $S_i \in L_k$ into two segments T and U. Then, the error of L_{k+1} is:

$$\begin{aligned}
\text{err}(L_{k+1}) &= \text{err}(L_k) - \frac{|S_i|}{m}\text{err}(S_i) + \frac{|T|}{m}\text{err}(T) + \frac{|U|}{m}\text{err}(U) \\
&= \text{err}(L_k) + \frac{|S_i|}{m}\mu_{S_i}^2 - \frac{|T|}{m}\mu_T^2 - \frac{|U|}{m}\mu_U^2 \\
&= \text{err}(L_k) - \frac{|T| \cdot |U|}{|S_i| \cdot m}(\mu_T - \mu_U)^2.
\end{aligned}$$

where the second equality is due to Equation 5 and the fact that the $\sum_j \mu_j^2$ is fixed no matter the segment approximation that is used, while last equality comes from the fact that $|T| + |U| = |S_i|$ and $\mu_{S_i} = (|T|\mu_T + |U|\mu_U)/|S_i|$. We use the above expression to define the notion of *improvement potential*. The *improvement potential* of a segment $S_i \in L_k$ is the minimization of the error of L_{k+1} obtained by splitting S_i into T and U. Thus, the *improvement potential* of segment S_i relative to T and U is

$$\Delta(S_i, T, U) = \frac{|T| \cdot |U|}{|S_i| \cdot m}(\mu_T - \mu_U)^2 \tag{3}$$

For any segment $S_i = (I_i, \eta_i)$, every group in the interval I_i except the last one can be chosen to be the *split group* (see Section 2.2). Therefore, there are $|S_i| - 1$ possible ways to choose $T, U \subseteq S_i$ such that $T \cup U = S$. The *split group* maximizing the *improvement potential* of S_i, minimizes the error of L_{k+1}. The maximum

improvement potential of a segment is expressed as follows:

$$\Delta^\star(S_i) = \max_{T,U \subseteq S_i} \Delta(S_i, T, U) = \max_{T,U \subseteq S_i} \frac{|T| \cdot |U|}{|S_i| \cdot m} (\mu_T - \mu_U)^2$$

Lastly, we denote the *improvement potential* of a given L_{k+1} by $\phi(L_{k+1}, S_i, T, U)$, where $\phi(L_{k+1}, S_i, T, U) = \Delta(S_i, T, U)$. Therefore, the maximum *improvement potential* of L_{k+1}, $\phi^\star(L_{k+1}) = \max_{S_i \subseteq L_k} \Delta^\star(S_i)$. When the means of the distributions are known, at iteration $(k+1)$, the optimal algorithm simply selects the refinement corresponding to $\phi^\star(L_{k+1})$, which is the segment approximation with the maximum improvement potential.

3.2 Case 2: The Online-Sampling Scenario

We now consider the case where the means $\mu_1, \ldots, \mu_m$ are *unknown* and we estimate each mean by drawing samples. Similar to the previous case, we want to identify a segment $S_i \in L_k$ such that splitting S_i into T and U results in the maximum *improvement potential*. We will first describe our approach for a given iteration assuming samples have been taken, then we will describe our approach for selecting samples, following which, we will establish a lower-bound.

3.2.1 Selecting the Refinement Given Samples

We first describe our approach, and then the justification.

Approach. As in the previous setting, our goal is to identify the refinement with the maximum improvement potential. Unfortunately, since the means are unknown, we cannot measure the exact improvement potential, so we minimize the *empirical* improvement potential based on samples seen so far. Analogous to the previous section, we simply pick the refinement that maximizes $\frac{|T| \cdot |U|}{|S_i| \cdot m} (\widetilde{\mu}_T - \widetilde{\mu}_U)^2$, where the $\widetilde{\mu}$s are the empirical estimates of the means.

Justification. At iteration $(k+1)$, we first draw samples from the distributions $D_1, \ldots, D_m$ to obtain the estimated means $\tilde{\mu}_1, \ldots, \tilde{\mu}_m$. For each $S_i \in L_k$, we set its value to $\eta_i = \widetilde{\mu}_{S_i} = \sum_{j \in S_i} \tilde{\mu}_j / |S_i|$, which we call the *estimated mean* of S_i. For any refinement L_{k+1} of L_k, we then let the *estimated improvement potential* of L_{k+1} be

$$\widetilde{\phi}(L_{k+1}, S_i, T, U) = \frac{|T|}{m} \widetilde{\mu}_T^2 + \frac{|U|}{m} \widetilde{\mu}_U^2 - \frac{|S|}{m} \widetilde{\mu}_{S_i}^2 = \frac{|T| \cdot |U|}{|S_i| \cdot m} (\widetilde{\mu}_T - \widetilde{\mu}_U)^2$$

For simplicity we denote $\phi(L_{k+1}, S_i, T, U)$ and $\widetilde{\phi}(L_{k+1}, S_i, T, U)$ as $\phi(L_{k+1})$ and $\widetilde{\phi}(L_{k+1})$, respectively.

Our goal is to get a guarantee for $\mathrm{err}(L_{k+1})$ is relative to $\mathrm{err}(L_k)$. Instead of a guarantee on the actual error err, for which we would need to know the means of the distributions, our guarantee is instead on a new quantity, err', which we define to be the *empirical* error. Given $S_i = (I_i, \eta_i)$ and Equation 5, err' is defined as follows: $\mathrm{err}'(S_i) = \frac{1}{|S_i|} \sum_{j \in S_i} \mu_j^2 - \eta_i^2$. Although $\mathrm{err}'(S_i)$ is not equal to $\mathrm{err}(S_i)$ when $\eta_i \neq \mu_S$, $\mathrm{err}'(S_i)$ converges to $\mathrm{err}(S_i)$ as η_i gets closer to μ_S (i.e., as more samples are taken). Similarly, the error of k-segment approximation $\mathrm{err}'(L_k)$ converges to $\mathrm{err}(L_k)$. We show experimentally (see Section 5) that optimizing for $\mathrm{err}'(L_{k+1})$ gives us a good solution of $\mathrm{err}(L_{k+1})$ itself.

To derive our guarantee on err', we need one more piece of terminology. At iteration $(k+1)$, we define $T(I, \eta)$, where $I = [p, q]$ and $1 \leq p \leq q \leq m$ to be a *boundary segment* if either p or q is a *split group* in L_k. In other words, at the iteration $(k+1)$, all the segments in L_k and all the segments that may appear in L_{k+1} after splitting a segment are called boundary segments. Next, we show that if we estimate the boundary segments accurately, then we can find a split which is very close to the best possible split.

Theorem 1. *If for every boundary segment T of the k-segment approximation L_k, we obtain an estimate $\widetilde{\mu}_T$ of the mean μ_T that satisfies $\left|\widetilde{\mu}_T^2 - \mu_T^2\right| \leq \frac{\epsilon m}{6|T|}$, then the refinement $L_{k+1}^\dagger$ of L_k that maximizes the estimated value $\widetilde{\phi}(L_{k+1}^\dagger)$ will have error that exceeds the error of the best refinement L_{k+1}^* of L_k by at most $\mathrm{err}'(L_{k+1}^\dagger) - \mathrm{err}'(L_{k+1}^*) \leq \epsilon$.*

Proof. The *estimated improvement potential* of the refinement L_{k+1} satisfies

$$\begin{aligned}
&|\widetilde{\phi}(L_{k+1}) - \phi(L_{k+1})| \\
&\leq \left| \frac{|S|}{m} (\widetilde{\mu}_S^2 - \mu_S^2) \right| + \left| \frac{|T|}{m} (\widetilde{\mu}_T^2 - \mu_T^2) \right| + \left| \frac{|U|}{m} (\widetilde{\mu}_U^2 - \mu_U^2) \right| \\
&\leq \frac{\epsilon}{2}.
\end{aligned}$$

Together this inequality, the identity $\mathrm{err}(L_{k+1}) = \mathrm{err}(L_k) - \phi(L_{k+1})$, and the inequality $\phi(L_{k+1}) \leq \phi(L_{k+1}^\dagger)$ imply that

$$\begin{aligned}
&\mathrm{err}'(L_{k+1}^\dagger) - \mathrm{err}'(L_{k+1}^*) \\
&= \phi(L_{k+1}^*) - \phi(L_{k+1}^\dagger) \\
&= \phi(L_{k+1}^*) - \widetilde{\phi}(L_{k+1}^*) + \widetilde{\phi}(L_{k+1}^*) - \phi(L_{k+1}^\dagger) \\
&\leq \phi(L_{k+1}^*) - \widetilde{\phi}(L_{k+1}^*) + \widetilde{\phi}(L_{k+1}^\dagger) - \phi(L_{k+1}^\dagger) \\
&\leq \epsilon.
\end{aligned}$$

□

3.2.2 Determining the Sample Complexity

To achieve the guarantee for Theorem 1, we need to retrieve a certain number of samples from each of the distributions $D_1, \ldots, D_m$. We now describe our approach for drawing samples.

Approach. Perhaps somewhat surprisingly, we find that we need to draw a constant $C = \left\lceil \frac{288\, a\, \sigma^2}{\epsilon^2 m} \ln\left(\frac{4m}{\delta}\right) \right\rceil$ from each distribution D_i to satisfy the requirements of Theorem 1. (We will define these parameters subsequently.) Thus, our sampling approach is remarkably simple—and is essentially *uniform sampling*—plus, as we show in the next subsection, other approaches cannot provide significantly better guarantees. What is not simple, however, is showing that taking C samples allows us to satisfy the requirements for Theorem 1. Another benefit of uniform sampling is that we can switch between showing our segment approximations or showing the actual running estimates of each of the groups (as in online aggregation [20])—for the latter purpose, uniform sampling is trivially optimal.

Justification. To justify our choice, we assume that the data is generated from a *sub-Gaussian distribution* [45]. Sub-Gaussian distributions form a general class of distributions with a strong tail decay property, with its tails decaying at least as rapidly as the tails of a Gaussian distribution. This class encompasses Gaussian distributions as well as distributions where extreme outliers are rare—this is certainly true when the values derive from real observations that are bounded. We validate this in our experiments. In particular, any Gaussian distribution with variance σ^2 is sub-Gaussian with parameter σ^2. Therefore, we represent the distributions $D_1, \ldots, D_m$ by m sub-Gaussian distributions with mean μ_i and sub-Gaussian parameter σ.

Given this generative assumption, we can determine the number of samples required to obtain an estimate with a desired accuracy using Hoeffding's inequality [21]. Given m independent random samples $x_1, \ldots, x_m$ with sub-Gaussian parameter σ_i^2 and mean μ_i

$$\Pr\left[\left| \sum_{i=1}^{m} (x_i - \mu_i) \right| > t \right] \leq 2 \exp\left(-\frac{t^2}{2 \sum_{i=1}^{m} \sigma_i^2} \right).$$

Given Hoeffding's inequality along with the union bound, we can derive an upper bound on the number of samples we need to estimate the mean μ_i of each x_i.

Lemma 1. *For a fixed $\delta > 0$ and a k-segment approximation of the distributions $D_1, \ldots, D_m$ represented by m independent random samples $x_1, \ldots, x_m$ with sub-Gaussian parameter σ^2 and mean*

$\mu_i \in [0,a]$ *if we draw* $C = \left\lceil \frac{288\,a\,\sigma^2}{\epsilon^2 m} \ln\left(\frac{4m}{\delta}\right) \right\rceil$ *samples uniformly from each* x_i*, then with probability at least* $1-\delta$, $\left|\widetilde{\mu}_T^2 - \mu_T^2\right| \leq \frac{\epsilon m}{6|T|}$ *for every boundary segment* T *of* L_k.

Proof. Fix any boundary segment T contained in the segment $S \in L_k$. Then, we draw samples $x_{i,1}x_{i,2},\ldots,x_{i,C}$ uniformly from $x_i \in T$, then

$$\begin{aligned}\widetilde{\mu}_T - \mu_T &= \frac{1}{C|T|}\sum_{i\in T}\sum_{j=1}^{C} x_{i,j} - \frac{1}{|T|}\sum_{i\in T}\mu_i \\ &= \frac{1}{C|T|}\sum_{i\in T}\sum_{j=1}^{C}(x_{i,j}-\mu_i).\end{aligned}$$

x_i's are sub-Gaussian random variables with parameter σ^2. Therefore,

$$\begin{aligned}\Pr\left[|\widetilde{\mu}_T^2 - \mu_T^2| > \tfrac{\epsilon m}{6\,|T|}\right] &= \Pr\left[|\widetilde{\mu}_T - \mu_T|\,(\widetilde{\mu}_T + \mu_T) > \tfrac{\epsilon m}{6\,|T|}\right] \\ &\leq \Pr\left[|\widetilde{\mu}_T - \mu_T| > \tfrac{\epsilon m}{12\,a\,|T|}\right] \\ &= \Pr\left[\left|\sum_{i\in T}\sum_{j=1}^{C}(x_{i,j}-\mu_i)\right| > \tfrac{C\,\epsilon m}{12\,a}\right] \\ &\leq 2\exp\left(-\tfrac{C\,\epsilon^2 m^2}{288a^2\,|T|\,\sigma^2}\right) \leq \tfrac{\delta}{2m}\end{aligned}$$

By the union bound, the probability that one of the $2m$ boundary segments has an inaccurate estimate is at most δ. □

3.2.3 Deriving a Lower bound

We can derive a lower bound for the sample complexity of any algorithm for Problem 1:

Theorem 2. *Say we have a dataset* D *of* m *groups with means* $(\mu_1,\mu_2,\ldots,\mu_m)$*. Assume there exists an algorithm* $\mathcal{A}$ *that finds a* k*-segment approximation which is* ϵ*-close to the optimal* k*-segment approximation with probability 2/3. For sufficiently small* ϵ*,* $\mathcal{A}$ *draws* $\Omega(\sqrt{k}/\epsilon^2)$ *samples.*

The theorem states that any algorithm that outputs a k-segment approximation of which is ϵ-close to a dataset has to draw $O(\sqrt{k}/\epsilon^2)$ samples from the dataset. To show this, at a high level, we pick a dataset as the "hard case" and show that any algorithm that draws $o(\sqrt{k}/\epsilon^2)$ samples cannot estimate the means of the many of the segments accurately. Therefore, the output has error more than ϵ.

Proof. Assume for contradiction that $\mathcal{A}$ uses $o(k\epsilon^2)$ samples. We build a datasets randomly on m groups, namely D, and show that $\mathcal{A}$ cannot find ϵ-close k-segment approximations for D with high probability, which contradicts our assumption.

Let ϵ' be equal to 217ϵ. For sufficiently small ϵ, we can assume ϵ' is at most 1/4. Without loss of generality, assume k is even and m is a multiple of $3k/2$. Otherwise, increase k and m by adding dummy groups and this does not affect our calculation by more than constant factors. We build D via the following process: First, we partition the m groups into $k/2$ segments. Each of the segments contains $2m/k$ groups and we define t to be a third of that i.e. $t := (2m)/(3k)$. In each segment, all the samples from the first t groups have a fixed value $1/2-\epsilon'$. All the samples from the last t groups have a fixed value $1/2+\epsilon'$. To decide about the value of the samples from the middle groups, we randomly choose the mean p to be either $1/2-\epsilon'$ or $1/2+\epsilon'$ each with probability a half. In particular, a sample from the middle t groups is a Bernoulli random variable which is one (or zero) with probability p (or $1-p$). More formally, we define the distribution D_i over the group i as below

$$D(i) = \begin{cases} \text{Fixed dist. with value } 1/2-\epsilon' & \text{if } \lceil i/t\rceil = 1 \pmod 3. \\ \text{Bernoulli dist. with mean } p_{\lceil 2i/k\rceil} & \text{if } \lceil i/t\rceil = 2 \pmod 3. \\ \text{Fixed dist. with value } 1/2+\epsilon' & \text{if } \lceil i/t\rceil = 0 \pmod 3. \end{cases}$$

where $p_1,p_2,\ldots,p_{k/2}$ are $k/2$ random variables that are picked from $\{1/2-\epsilon', 1/2+\epsilon'\}$ each with probability a half. It is not hard to see that any D is a k-flat segment approximation, so the error of the optimal k-segment approximation is zero. Therefore, $\mathcal{A}$ outputs a segment approximation, namely L_o, which is ϵ-close to the D with probability 2/3. For the segment j, we define p_j (respectively $\hat{p}_j$) to be the average mean of the middle t groups of the j-th segment that D (respectively L_o) assigns to them. In other words, $p_j = \sum_{i=1}^{t} D(3\,(j-1)\,t+t+i)/t$ (respectively $\hat{p}_j = \sum_{i=1}^{t} L_o(3\,(j-1)\,t+t+i)/t$). With probability 2/3, we have

$$\begin{aligned}\epsilon &\geq \mathrm{err}(D,L_o) = \frac{1}{m}\sum_{i=1}^{m}\left(D(i)-L_o(i)\right)^2 \\ &\geq \frac{1}{m}\sum_{j=1}^{k/2}\sum_{i=1}^{t}\left(D\left(3\,(j-1)\,t+t+i\right) - L_o\left(3\,(j-1)\,t+t+i\right)\right)^2 \\ &= \frac{t}{m}\sum_{j=1}^{k/2}(p_j-\hat{p}_j)^2 \geq \frac{2}{3k^2}\left(\sum_{j=1}^{k/2}|p_j-\hat{p}_j|\right)^2\end{aligned}$$

using the Cauchy-Schwarz inequality.

To reach a contradiction, we show that we cannot estimate many of the p_j's accurately. Since we assumed that $\mathcal{A}$ draws $o(k/\epsilon^2)$ samples, there are at least $k/4$ segments that $o(1/\epsilon^2)$ samples are drawn from. In the following lemma, we show that we cannot estimate $\hat{p}_j$'s in these segments with high probability.

Lemma 2. *Assume Algorithm* $\mathcal{B}$ *draws* $o(1/\epsilon'^2)$ *samples from a distribution over* $\{0,1\}$ *with an unknown bias* p *which is* $1/2+\epsilon'$ *or* $1/2-\epsilon'$*.* $\mathcal{B}$ *outputs* $\hat{p}$ *as an estimation of* p*. With probability at least 1/3,*

$$|\hat{p}-p| \geq \epsilon'.$$

Proof. We reduce the problem of estimating p, to the problem of distinguishing two distributions D_1 and D_2 over $\{0,1\}$ which have average value of $1/2+\epsilon'$ and $1/2-\epsilon'$ respectively. The reduction goes as follows: Algorithm $\mathcal{B}'$, runs B to find $\hat{p}$. If $\hat{p}$ is greater than 1/2, then $\mathcal{B}'$ outputs D_1. Otherwise, it outputs D_2. It is clear that $\mathcal{B}'$ is correct if and only if $|\hat{p}-p| \geq \epsilon'$.

In Theorem 4.7, Chapter 4 of Yossef et al. [12], it has been shown that any hypothesis tester that distinguishes between D_1 and D_2 with probability 2/3 must use at least $\Omega(1/h^2(D_1,D_2))$ samples where $h(D_1,D_2)$ is the Hellinger distance between D_1 and D_2. The Hellinger distance between D_1 and D_2 over a finite set $\mathcal{X}$ (in our case $\mathcal{X}=\{0,1\}$) is defined by Cam et al. [14] as follows:

$$\begin{aligned}h(D_1,D_2) &:= \sqrt{\tfrac{1}{2}\sum_{x\in\mathcal{X}}\left(\sqrt{D_1(x)}-\sqrt{D_2(x)}\right)^2} \\ &= \sqrt{1-\sum_{x\in\mathcal{X}}\sqrt{D_1(x)D_2(x)}} \\ &= \sqrt{1-2\sqrt{1/4-\epsilon'^2}} = \sqrt{1-\sqrt{1-4\epsilon'^2}} \\ &= \sqrt{\frac{4\epsilon'^2}{1+\sqrt{1-4\epsilon'^2}}} \geq \sqrt{2}\epsilon'\end{aligned}$$

Since $\mathcal{B}'$ draws $o(1/\epsilon'^2)$ samples, it cannot output the correct answer with probability at least 2/3. Therefore, with probability at least 1/3, the error of $\hat{p}$ is at least ϵ'. □

Let r be the number of segments j such that we draw $o(1/\epsilon^2)$ samples from them but $|\hat{p}_j - p_j|$ is less than ϵ . Using the lemma above, $\mathrm{E}[r]$ is at least $k/6$. By the Chernoff bound,

$$\Pr\left[r < \frac{k}{12}\right] \leq \Pr\left[r < \left(1 - \frac{1}{2}\right) \cdot \mathrm{E}[r]\right] \leq e^{-k/48} \leq \frac{1}{6}$$

where the last inequality true for sufficiently large k. Thus, with probability $5/6$ more than $k/12$ has error of at least ϵ'. This implies that $\mathrm{err}(D, L_0) \geq \epsilon'/216 > \epsilon$ which contradicts our assumption. □

3.3 The *ISplit* Algorithm

We now present our incrementally improving visualization generation algorithm *ISplit*. Given the parameters ϵ and δ, *ISplit* maintains the same guarantee of error (ϵ) in generating the segment approximations in each iteration. Theorem 1 and Lemma 1 suffice to show that the *ISplit* algorithm is a greedy approximator, that, at each iteration identifies a segment approximation that is at least ϵ close to the best segment approximation for that iteration.

Data: X_a, Y, δ, ϵ
Start with the 1-segment approximator $L = (L_1)$.
for $k = 2, \ldots, m$ **do**
  $L_{k-1} = (S_1, \ldots, S_{k-1})$.
  for *each* $S_i \in L_{k-1}$ **do**
    for *each* *group* $q \in S_p$ **do**
      Draw C samples uniformly. Compute mean $\tilde{\mu}_q$
    end
    Compute $\widetilde{\mu}_{S_i} = \frac{1}{|S_i|} \sum_{q \in S_i} \tilde{\mu}_q$
  end
  Find $(T, U) = \mathrm{argmax}_{i;\, T, U \subseteq S_i} \frac{|T| \cdot |U|}{|S_i| \cdot m} (\widetilde{\mu}_T - \widetilde{\mu}_U)^2$.
  Update $L_{k+1} = S_1, \ldots, S_{i-1}, T, U, S_{i+1}, \ldots, S_k$.
end

Algorithm 1: *ISplit*

The parameter ϵ is often difficult for end-users to specify. Therefore, in INCVISAGE, we allow users to instead explicitly specify an expected *time budget* per iteration, B—as explained in Section 2.3, the number of samples taken determines the time taken per iteration, so we can reverse engineer the number of samples from B. So, given the sampling rate f of the sampling engine (see Section 2.1) and B, the sample complexity per iteration per group is simply $C = B \times f/m$. Using Lemma 1, we can compute the corresponding ϵ. Another way of setting B is to use standard rules of thumb for interactivity (see Section 2.3).

3.4 Tuning ϵ Across Iterations

So far, we have considered only the case where the algorithm takes the same number of samples C per group across iterations and does not reuse the samples across different iterations. For any given iteration, this provides us with an ϵ-guarantee for the error relative to the best segment approximation. If we instead reuse the samples drawn in previous iterations, the error at iteration k is $\epsilon_k^2 = \frac{288\, a\, \sigma^2}{mkC} \ln\left(\frac{4m}{\delta}\right)$, where $k\,C$ is the total number of samples drawn so far. Therefore, the decrease in the error up to iteration k, ϵ_k, from the error up to iteration $(k-1)$, ϵ_{k-1}, is $\sqrt{(k-1)/k}$, where $\epsilon_k = \sqrt{(k-1)/k}\,\epsilon_{k-1}$. This has the following effect: later iterations both take the same amount of time as previous ones, and produce only minor updates of the visualization.

Sampling Approaches. This observation indicates that we can obtain higher interactivity with only a small effect on the accuracy of the approximations by considering variants of the algorithm that decrease the number of samples across iterations instead of drawing the same number of samples in each iteration. We consider three natural approaches for determining the number of samples we draw at each iteration: linear decrease (i.e., reduce the number of samples by β at each iteration), geometric decrease (i.e., divide the number of samples by α at each iteration), and all-upfront (i.e., non-interactive) sampling. To compare these different approaches to the constant (uniform) sampling approach and amongst each other, we first compute the total sample complexity, interactivity, and error guarantees of each of them as a function of their other parameters. Letting T_k denote the total number of samples drawn in the first k iterations, the *interactivity* of a sampling approach defined in Section 2.3 can be written as: $\lambda = \frac{\sum_{k=1}^{m} T_k}{k'}$, where k' is the number of iterations we take non-zero samples. The error guarantees we consider are, as above, the average error over all iterations. This error guarantee is

$$\mathrm{err} = \sum_{i=1}^{m} \frac{\epsilon_k^2}{m} = \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k} \ln\left(\tfrac{4m}{\delta}\right) = \frac{288\, a\, \sigma^2}{m^2} \ln\left(\tfrac{4m}{\delta}\right) \sum_{i=1}^{m} \frac{1}{T_k}$$

We provide evidence that the estimated err and λ are similar to err and λ on real datasets in Section 5.5.2. We are now ready to derive the expressions for both error and interactivity for the sampling approaches mentioned earlier.

3.4.1 Expressions for Error and Interactivity

In this section we derive the analytical expressions of both interactivity and error for all of these four approaches (see Table 1). In particular, for succinctness, we have left the formulae for Error for geometric and linear decrease as is without simplification; on substituting T_k into the expression for Error, we obtain fairly complicated formulae with dozens of terms, including multiple uses of the q-digamma function Ψ [25]—for the geometric decrease case, the Euler-Mascheroni constant [47]—for the linear decrease case, and the Harmonic number H_m—for the constant sampling case.

Constant sampling. The *constant sampling* approach is the one described above where we draw N_1 samples in the first iteration and in all subsequent iterations as well. We denote this approach by U_{N_1}. With this approach, the total number of samples drawn in the first k iterations is $T_k^{(\mathrm{U}_{N_1})} = kN_1$ and the error guarantee satisfies

$$\mathrm{err}^{(\mathrm{U}_{N_1})} = \frac{288\, a\, \sigma^2}{m^2} \ln\left(\frac{4m}{\delta}\right) \sum_{i=1}^{m} \frac{1}{k\, N_1} = \frac{288\, a\, \sigma^2 H_m}{m^2 N_1} \ln\left(\frac{4m}{\delta}\right),$$

where H_m is the m-th Harmonic number. Finally, the interactivity of the constant sampling approach is

$$\lambda^{(\mathrm{U}_{N_1})} = \frac{\sum_{k=1}^{m} kN_1}{k'} = \frac{N_1\,(m+1)}{2}.$$

Linear decrease. In the linear decrease approach, we draw N_1 samples in the first iteration, and draw $N_k = N_{k-1} - \beta$ samples for some fixed parameter $\beta \geq 0$ in each subsequent iterations. We denote this sampling approach by $\mathrm{L}_{N_1,\beta}$. The total number of samples drawn in the first k iterations of this approach is $T_k^{(\mathrm{L}_{N_1,\beta})} = \sum_{i=1}^{k} N_i = N_1 k - \frac{k\,(k-1)\,\beta}{2}$. The guarantee on the average error per iteration for this approach is

$$\mathrm{err}^{(\mathrm{L}_{N_1,\beta})} = \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k} \ln\left(\tfrac{4m}{\delta}\right)$$
$$= \frac{288\, a\, \sigma^2\,(\alpha-1)\left(-\Psi^{(0)}(m - 2N_1/\beta)\Psi^{(0)}(-2N_1/\beta) + \Psi^{(0)}(m+1) + \gamma\right)}{m^2\,(N_1 + \beta/2)} \ln\left(\tfrac{4m}{\delta}\right).$$

where the $\Psi^{(n)}(x)$ is the nth derivative of the digamma function [25] and γ is the Euler-Mascheroni constant [47]. The interactivity of the linear decrease approach is

$$\lambda^{(\mathrm{L}_{N_1,\beta})} = \sum_{k=1}^{k'} \frac{N_k\,(m-k+1)}{k'} = \sum_{k=1}^{k'} \frac{(N_1 - \beta(k-1)) \cdot (m-k+1)}{k'}$$
$$= N_1 \left(m - \frac{k'-1}{2}\right) + \frac{\beta}{6}\left[2k'^2 - 3k' + 1 - 3mk' + 3m\right].$$

Geometric decrease. In the geometric decrease approach, we draw N_1 samples in the first iteration and draw $N_k = N_{k-1}/\alpha$ samples for some fixed parameter $\alpha > 1$ in each subsequent iterations. We denote this sampling approach by $\mathrm{G}_{N_1,\alpha}$. This ap-

Approach	decrease parameter	T_k	Error	Interactivity
Linear Decrease	β	$N_1 k - \frac{k\,(k-1)\,\beta}{2}$	$A \sum_{k=1}^{m} \frac{1}{T_k}$	$N_1 \left(m - \frac{k'-1}{2}\frac{N_1}{m}\right) + \frac{\beta}{6}\left[2k'^2 - 3k' + 1 - 3mk' + 3m\right]$
Geometric Decrease	α	$N_1 \frac{\alpha^k - 1}{\alpha^{k-1}(\alpha-1)}$	$A \sum_{k=1}^{m} \frac{1}{T_k}$	$\frac{N_1}{m}\left[\frac{m\,(\alpha-1)\,\alpha^m + \alpha^m - 1}{(\alpha-1)^2\,\alpha^{m-1}}\right]$
Constant Sampling	$\alpha = 1$, or $\beta = 0$	kN_1	$\frac{A \cdot H_m}{N_1}$	$\frac{N_1\,(m+1)}{2}$
All-upfront	—	$T_{k=1} = N_1$ and $T_{k>1} = 0$	$\frac{A \cdot m}{N_1}$	mN_1

Table 1: Expressions for error and interactivity for different sampling approaches. Here, $A = \frac{288\,a\,\sigma^2}{m^2} \ln\left(\frac{4m}{\delta}\right)$

proach draws $N_k = \frac{N_1}{\alpha^{k-1}}$ samples at iteration k, for a total of $T_k^{(\mathrm{G}_{N_1,\alpha})} = \sum_{i=1}^{k} N_i = N_1 \frac{\alpha^k - 1}{\alpha^{k-1}(\alpha-1)}$ samples in the first k iterations. The error guarantee of this approach is

$$\begin{aligned}\mathrm{err}^{(\mathrm{G}_{N_1,\alpha})} &= \sum_{i=1}^{m} \frac{288\,a\,\sigma^2\alpha^{k-1}(\alpha-1)}{m^2 N_1 \alpha^k - 1} \ln\left(\tfrac{4m}{\delta}\right) \\ &= \frac{288\,a\,\sigma^2\,(\alpha-1)\left(\Psi_\alpha^{(0)}(m+1) - \Psi_\alpha^{(0)}(1)\right)}{m^2\,N_1\,\alpha\,\ln\alpha} \ln\left(\tfrac{4m}{\delta}\right)\end{aligned}$$

where $\Psi_\alpha^{(n)}(x)$ is the q-digamma function. The interactivity of the geometric decrease approach is

$$\begin{aligned}\lambda^{(\mathrm{G}_{N_1,\alpha})} &= \sum_{k=1}^{k'} \frac{N_k\,(m-k+1)}{k'} \\ &= \frac{N_1(m+1)}{k'} \sum_{k=1}^{k'} \frac{1}{\alpha^{k-1}} - \frac{N_1}{k'} \sum_{k=1}^{k'} \frac{k}{\alpha^{k-1}} \\ &= \frac{N_1(m+1)}{k'} \frac{\alpha^{k'}-1}{\alpha^{k'-1}(\alpha-1)} - \frac{N_1}{k'} \frac{\alpha^{k'+1} - \alpha(k'+1) + k'}{\alpha^{k'-1}(\alpha-1)^2}.\end{aligned}$$

For the case where $k' = m$, then

$$\lambda^{(\mathrm{G}_{N_1,\alpha})} = \frac{N_1}{m}\left[\frac{m\,(\alpha-1)\,\alpha^m - \alpha^m + 1}{(\alpha-1)^2\,\alpha^{m-1}}\right].$$

All-upfront. Finally, the (non-interactive) *all-upfront* sampling approach is the one where we draw N_1 samples in the first iteration and no samples thereafter. Let (AU_{N_1}) denote this sampling approach. For any $k \geq 1$, the total sample complexity of this approach is $T_k^{(\mathrm{AU}_{N_1})} = N_1$. The error guarantee of this approach is

$$\mathrm{err}^{(\mathrm{AU}_{N_1})} = \frac{288\,a\,\sigma^2}{m^2} \ln\left(\tfrac{4m}{\delta}\right) \sum_{i=1}^{m} \frac{1}{N_1} = \frac{288\,a\,\sigma^2}{mN_1} \ln\left(\tfrac{4m}{\delta}\right),$$

and its interactivity is

$$\lambda^{(\mathrm{AU}_{N_1})} = \frac{\sum_{k=1}^{m} N_1}{k'} = m\,N_1.$$

3.4.2 Comparing the Sampling Approaches

We now compare different sampling approaches based on the expressions of error and interactivity obtained in Section 3.4.1. We show that among the four sampling approaches, geometric decrease is the most desirable approach in order to optimize for both error and interactivity. To do so, we first discount the all-upfront sampling approach by showing that this approach has a strictly worse error guarantee than constant sampling—the sampling approach proposed in Section 3.2.2. We then obtain the optimal values in terms of interactivity for the decrease parameter of the geometric (α) and linear (β) decrease approaches. Furthermore, we show that given the initial number of samples, the interactivity of geometric decrease approach with the optimal decrease parameter is strictly better than the interactivity of the linear decrease and constant sampling approaches. Finally, we suggest a theoretically optimal range of the geometric decrease parameter that leads to a pareto frontier along which users can optimize for either error or interactivity.

All-upfront VS Constant Sampling. Clearly, if we compare all-upfront sampling with the constant sampling approach with the same number of initial samples N_1, then the constant sampling approach has smaller error. In fact, more is true: even if we allow the all-upfront to take *more* samples in the initial iteration so that it has the same interactivity measure as the constant sampling approach, it still has a strictly worse error guarantee.

Theorem 3. *If for a setting of parameters, a constant sampling approach and an all-upfront sampling approach have the same interactivity, then the error of constant sampling is strictly less than the error of all-upfront sampling.*

Proof. Assume that we have two sampling approaches AU_{N_1} and $\mathrm{U}_{N_1'}$ with the same interactivity. Thus, we have the following relationship between N_1 and N_1'.

$$m\,N_1 = \lambda^{(\mathrm{AU}_{N_1})} = \lambda^{(\mathrm{U}_{N_1'})} = \frac{N_1'\,(m+1)}{2} \leq m\,N_1'.$$

Given the equation, we have

$$\begin{aligned}\mathrm{err}^{(\mathrm{AU}_{N_1})} &= \frac{288\,a\,\sigma^2}{mN_1} \ln\left(\tfrac{4m}{\delta}\right) \\ &\geq \frac{288\,a\,\sigma^2}{mN_1'} \ln\left(\tfrac{4m}{\delta}\right) \\ &\geq \frac{m}{H_m} \cdot \mathrm{err}^{(\mathrm{U}_{N_1'})}.\end{aligned}$$

Note that $H_m = O(\log n)$. Hence, given the same interactivity, the non-incremental sampling has a large error compare to the uniform sampling. □

Optimal interactivity and decrease parameters. We now show that for both linear and geometric decrease approaches, the optimal interactivity is obtained for an explicit choice of decrease parameter that depends only on the initial number of samples and the total number of iterations. To do so, we prove the following two lemmas (lemma 3 and 4).

Lemma 3. *In geometric sampling with a fixed* N_1, $\alpha^* = (N_1 - 1)^{1/(m-1)}$ *has the optimal interactivity and it has smaller error than all of the geometric sampling approaches with* $\alpha > \alpha^*$.

Proof. If $\alpha > \alpha^*$, it is clear that we do not draw any sample at the last iteration, and therefore, $k' < m$. However, when $\alpha \leq \alpha^*$, then $k' = m$. Hence,

$$\lambda^{(\mathrm{G}_{N_1,\alpha})} = \frac{N_1}{m}\left[\frac{m\,(\alpha-1)\,\alpha^m - \alpha^m + 1}{(\alpha-1)^2\,\alpha^{m-1}}\right].$$

Then,

$$\begin{aligned}\frac{\partial \lambda^{(\mathrm{G}_{N_1,\alpha})}}{\partial \alpha} &= \frac{\alpha^{-m}\left((m+1)\alpha^m - (m-1)\alpha^{m+1} + m - 1 - \alpha(1+m)\right)}{(\alpha-1)^3} \\ &= \frac{\alpha^{-m}(\alpha^m(2-(m-1)(\alpha-1))) - (\alpha-1)(m+1) - 2)}{(\alpha-1)^3}.\end{aligned}$$

By writing the binomial approximation for $\alpha^m = (1-(1-\alpha))^m$. It is not hard to see that the derivative is negative for $\alpha \geq 1$ for sufficiently large m. Therefore, $\lambda^{(\mathrm{G}_{N_1,\alpha})}(\alpha^*)$ is the minimum among all α's in $(1, \alpha^*]$.

For $\alpha > \alpha^*$, we stop sampling after $k' = \frac{\log N_1}{\log \alpha}$ iterations. By replacing k' in the interactivity formula we can obtain an expression for $\lambda^{(\mathrm{G}_{N_1,\alpha})}$. On examining the derivative we find out that $\lambda^{(\mathrm{G}_{N_1,\alpha})}$ is again an increasing function of α. Therefore, α^* has the optimal interactivity.

Now, we show that $\mathrm{G}_{N_1,\alpha^*}$ has smaller error compared to $\mathrm{G}_{N_1,\alpha}$ for $\alpha > \alpha^*$. Suppose $\alpha > \alpha^*$. It is clear that

$$N_k^{(\mathrm{G}_{N_1,\alpha})} = \frac{N_1}{(\alpha)^{k-1}} \leq \frac{N_1}{(\alpha^*)^{k-1}} = N_k^{(\mathrm{G}_{N_1,\alpha^*})}.$$

Thus, for any $k \in [m]$, $T_k^{(\mathrm{G}_{N_1,\alpha})} \leq T_k^{(\mathrm{G}_{N_1,\alpha^*})}$. Therefore, we can see that the error of the $\mathrm{G}_{N_1,\alpha^*}$ is smaller than $\mathrm{G}_{N_1,\alpha}$.

$$\begin{aligned}\mathrm{err}^{(\mathrm{G}_{N_1,\alpha^*})} &= \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k^{(\mathrm{G}_{N_1,\alpha^*})}} \ln\left(\tfrac{4m}{\delta}\right) \\ &\leq \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k^{(\mathrm{G}_{N_1,\alpha})}} \ln\left(\tfrac{4m}{\delta}\right) \\ &= \mathrm{err}^{(\mathrm{G}_{N_1,\alpha})}.\end{aligned}$$

Thus, the proof is complete. □

Lemma 4. *In linear sampling with a fixed N_1, $\beta^* = (N_1 - 1)/(m-1)$ has the optimal interactivity and it has strictly smaller error than all of the linear sampling approaches with $\beta > \beta^*$.*

Proof. To compute the interactivity of the linear decrease approach, first we need to find k' based on β. If the sampling lasts for m iterations, we draw at least one sample in the last iteration. In other words, $N_1 - (m-1)\beta$ is at least one. Therefore, $\beta \leq \beta^* = (N_1 - 1)/(m-1)$ the number of iterations, k', is m. Then, the interactivity is

$$\lambda^{(\mathrm{L}_{N_1,\beta})} = N_1 \left(\frac{m+1}{2}\right) - \frac{\beta}{6}\left(m^2 - 1\right).$$

For sufficiently large m, $\lambda^{(\mathrm{L}_{N_1,\beta})}$ is decreasing with respect to β. Therefore, β^* has the optimal interactivity among β's in $[0, \beta^*]$.

For $\beta > \beta^*$, we stop sampling after $k' = \frac{N_1}{\beta}$ iterations. By replacing k' in the interactivity formula, we have

$$\lambda^{(\mathrm{L}_{N_1,\beta})} = \frac{N_1(m+1)}{2} - \frac{N_1^2}{6\beta} + \frac{(3m+1)\beta}{6}$$

It is not hard to see that $\frac{\partial \lambda^{(\mathrm{L}_{N_1,\beta})}}{\partial \beta}$ is positive and this function is increasing with respect to β. Therefore, β^* has the optimal interactivity among β's in $[\beta^*, \beta]$. Hence β^* has the optimal interactivity.

Now, we show that L_{N_1,β^*} has smaller error compared to $\mathrm{L}_{N_1,\beta}$ for $\beta > \beta^*$. Suppose $\beta > \beta^*$. It is clear that

$$N_k^{(\mathrm{L}_{N_1,\beta})} = N_1 - (k-1)\beta \leq N_1 - (k-1)\beta^* = N_k^{(\mathrm{L}_{N_1,\beta^*})}.$$

Thus, for any $k \in [m]$, $T_k^{(\mathrm{L}_{N_1,\beta})} \leq T_k^{(\mathrm{L}_{N_1,\beta^*})}$. Therefore, we can easily see error of the L_{N_1,β^*} is smaller than $\mathrm{L}_{N_1,\beta}$.

$$\begin{aligned}\mathrm{err}^{(\mathrm{L}_{N_1,\beta^*})} &= \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k^{(\mathrm{L}_{N_1,\beta^*})}} \ln\left(\tfrac{4m}{\delta}\right) \\ &\leq \sum_{i=1}^{m} \frac{288\, a\, \sigma^2}{m^2 T_k^{(\mathrm{L}_{N_1,\beta})}} \ln\left(\tfrac{4m}{\delta}\right) \\ &= \mathrm{err}^{(\mathrm{L}_{N_1,\beta})}.\end{aligned}$$

Thus, the proof is complete. □

Next, we show that given the initial samples N_1, geometric decrease has the optimal interactivity among the three interactive approaches (linear decrease and constant sampling being the other two).

Optimal Interactivity Given N_1. Our experimental results in Section 5.5 suggests that the geometric decrease approach with the optimal choice of parameter α^* has better error than not only the all-upfront approach but the linear decrease and constant sampling approaches as well. This remains an open question, but when we fix the initial sample complexity N_1 (which is proportional to the bound on the maximum time taken per iteration as provided by the user), we can show that geometric decrease with the right choice of parameter α does have the optimal interactivity among the three interactive approaches.

Theorem 4. *Given N_1, the interactivity of geometric decrease with parameter $\alpha^* = (N_1 - 1)^{1/(m-1)}$ is strictly better than the interactivity of any linear decrease approach and constant sampling.*

Proof. By Lemma 4, for a fixed N_1, the interactivity of $\mathrm{L}_{N_1,\beta}$ is minimum at $\beta^* = (N_1 - 1)/(m-1)$. In addition, the uniform sampling approach is a special case of linear decrease sampling approach when β is zero, and therefore has worse interactivity compared to L_{N_1,β^*}. Thus, it suffices to compare the interactivity of L_{N_1,β^*} and $\mathrm{G}_{N_1,\alpha^*}$ to conclude the theorem.

First, we prove that $N_k^{(\mathrm{L}_{N_1,\beta^*})}$ is at least $N_k^{(\mathrm{G}_{N_1,\alpha^*})}$. $N_k^{(\mathrm{G}_{N_1,\alpha^*})} = N_1/(\alpha^*)^{k-1}$ is a upward convex function with respect to k. Thus, the line segment between any two points of this function lies above it. This sampling approach takes N_1 samples in the first iteration and one sample in iteration m. Thus, $N_k^{(\mathrm{G}_{N_1,\alpha^*})}$ is below the segment that connects $(1, N_1)$ and $(m, 1)$. On the other hand $N_k^{(\mathrm{L}_{N_1,\beta^*})} = N_1 - (k-1)\beta^*$ is a linear function of k. Also, L_{N_1,β^*} takes N_1 samples in the first iteration and one sample in iteration m. Thus, $N_k^{(\mathrm{L}_{N_1,\beta^*})}$ is on the segment $(1, N_1)$ and $(m, 1)$. Therefore, $N_k^{(\mathrm{L}_{N_1,\beta^*})}$ is at least $N_k^{(\mathrm{G}_{N_1,\alpha^*})}$ and the equality happens only at $k = 1$ and $k = m$. Using the interactivity formula it is not hard to see

$$\begin{aligned}\lambda^{(\mathrm{G}_{N_1,\alpha^*})} &= \sum_{i=1}^{m} \frac{N_k^{(\mathrm{G}_{N_1,\alpha^*})}(m-k+1)}{m} \\ &< \sum_{i=1}^{m} \frac{N_k^{(\mathrm{L}_{N_1,\beta^*})}(m-k+1)}{m} = \lambda^{(\mathrm{L}_{N_1,\beta^*})}\end{aligned}$$

Therefore, the geometric decrease sampling approach, $\mathrm{G}_{N_1,\alpha^*}$, has better interactivity than the interactivity of any linear decrease sampling method.

In lemma 3, we prove that for fixed N_1 $\lambda^{(\mathrm{G}_{N_1,\alpha})}(\alpha^*)$ is the minimum among all α's in $(1, \alpha^*]$. Now, constant sampling is a special case of geometric decrease with parameter $\alpha = 1$. Therefore, given N_1, the geometric decrease sampling approach, $\mathrm{G}_{N_1,\alpha^*}$, has better interactivity than the interactivity of the constant sampling approach. This completes our proof. □

Next, we discuss how the optimal decrease parameter α^* contributes to a knee shaped error-interactivity trade-off curve.

Optimal α and Knee Region. As shown in Lemma 3, given N_1, we can compute the optimal decrease parameter α^*, such that any value of $\alpha > \alpha^*$ results in higher error and lesser interactivity (higher λ). This behavior results into the emergence of a knee region in the error-interactivity curve which is confirmed in our experimental results (see Section 5.5). Essentially, starting from $\alpha = 1$ any value $\alpha \leq \alpha^*$ has smaller error than any value $\alpha > \alpha^*$. Therefore, for any given N_1 there is an optimal region $[1, \alpha^*]$. For example, for $N_1 = 5000, 25000, 5000$, and 10000, the optimal range of α is $[1, 1.024]$, $[1, 1.028]$, $[1, 1.03]$, and $[1, 1.032]$, respectively. By varying α along the optimal region one can either optimize for error or interactivity. For example, starting with $\alpha = 1$ as $\alpha \to \alpha^*$ we trade accuracy for interactivity.

3.5 Extensions

The incrementally improving visualization generation algorithm described previously for simple queries with the AVG aggregation function can also be extended to support aggregations such as COUNT and SUM, as well as additional predicates (via a WHERE clause).

3.5.1 The COUNT *Aggregate Function*

Given that we know the total number of tuples in the database, estimating the COUNT aggregate function essentially corresponds to the problem of estimating the fraction of tuples τ_i that belong to each group i. Formally, $\tau_i = \frac{n_i}{\sum_{j=1}^{m} n_j}$ when n_j is the number of tuples in group j. We focus on the setting below when we only use our bitmap index. We note that approximation techniques

for COUNT queries have also been studied previously [7, 24, 26], for the case when no indexes are available. As we see below, we will also use the COUNT estimation as a subroutine for incrementally improving approximations to the SUM function in the setting where we don't know the group sizes.

Approach. Using our sampling engine, we can estimate the fractions τ_i by scanning the bitmap index for each group. When we retrieve another sample from group i, we can also estimate the number of tuples we needed to skip over to reach the tuple that belongs to group i—the indices allow us to estimate this information directly, providing us an estimate for τ_i, i.e., $\tilde{\tau}_i$.

Theorem 5. *With an expected total number of samples $C_{count} = m + \lceil \gamma^{-2} \ln(2m/\delta)/2 \rceil$, the $\tilde{\tau}_i, \forall i$ can be estimated to within a factor γ, i.e., $|\tilde{\tau}_i - \tau_i| \leq \gamma$ holds with probability at least $1 - \delta$.*

Essentially, the algorithm takes as many samples from each group until the index of the sample is $\geq \lceil \gamma^{-2} \ln(2m/\delta)/2 \rceil$. We show that overall, the expected number of samples is bounded above by C_{count}. Since this number is small, we don't even need to incrementally estimate or visualize COUNT.

Proof. To show this, we use repeated applications of Hoeffding's inequality and union bound, along with the analysis of a Bernoulli trial. Let $\theta_{i,j}$ be the index of the j-th sample from group i. Another way of interpreting $\theta_{i,j}$ is that among the first $\theta_{i,j}$ items in the dataset, j of them were from group i. This is equivalent to drawing $\theta_{i,j}$ samples in the bitmap where only j of them are from the group i. Thus, $j/\theta_{i,j}$ is an unbiased estimate for τ_i. Using the Hoeffding's inequality, we have that

$$\Pr\left[\left|\frac{j}{\theta_{i,j}} - \tau_i\right| > \gamma\right] \leq 2e^{-2\gamma^2\theta_{i,j}} \leq \frac{\delta}{m}$$

whenever $\theta_{i,j}$ is greater than or equal to $\theta_0 = \lceil \gamma^{-2} \ln(2m/\delta)/2 \rceil$. Therefore, if the $\theta_{i,j}$'s are big enough, we can assume we have a good estimation of τ_i's with probability $1 - \delta$.

In this approach, we do not query the dataset. However, we query the bitmap index. To reach $\theta_{i,j}$ that is greater than θ_0, we query the bitmap index to obtain $\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,j}$ until we see $\theta_{i,j}$ which is at least θ_0. Here, we compute the expected value of the number of queries where each query is a Bernoulli trial. We define a Bernoulli trial as follows: we draw an item from the dataset if the item belongs to group i then it is a success. Otherwise, it is a failure. We know that among the first $\theta_{i,j-1}$, $j-1$ of them were successful. Thus, we have

$$\mathrm{E}[j] = \mathrm{E}[j-1] + 1 \leq \mathrm{E}[\#\text{success in } \theta_0 \text{ trials}] + 1 = \tau_i\theta_0 + 1.$$

Therefore, for m groups, the expected number of queries to the bitmap index, $C_{count} = \theta_0 + m = m + \lceil \gamma^{-2} \ln(2m/\delta)/2 \rceil$. □

3.5.2 The SUM Aggregate Function

The problem of obtaining the segment approximations for SUM is similar to the AVG case—at each iteration $(k+1)$, a segment is split into two new segments to obtain the $k+1$-segment approximation L_{k+1} such that the *estimated improvement potential* is maximized. For the SUM problem, we define the estimated improvement potential as $\widetilde{\phi}^{+}$. In the online sampling scenario, we again obtain a guarantee for the empirical error of the k-segment approximation for SUM, err'^{+}. There are two settings we consider for the SUM aggregate function: when the group sizes (i.e., the number of tuples in each group) are known, and when they are unknown. For both cases we show that if we estimate the boundary segments accurately, then we can find a split which is very close to the best possible split. At iteration $(k+1)$, we define $T(I, \eta)$, where $I = [p, q]$ and $1 \leq p \leq q \leq m$ to be a *boundary segment* if either p or q is a *split group* in L_k (see Section 3.2).

Known Group Sizes. A simple variant of Algorithm 1 can also be used to approximate SUM in this case. Let n_i be the size of group i and $\kappa = \max_j n_j$. As in the original setting, the algorithm computes the estimate $\tilde{\mu}_i$ of the average of the group. Then $\tilde{s}_i = n_i\,\tilde{\mu}_i$ is an estimate on the sum of each group i, namely s_i, that is used in place of the average in the rest of the algorithm. We have:

Theorem 6. *Assume we have $C_i = \lceil \frac{288a^2\sigma^2 m n_i^2}{\epsilon^2\kappa^2} \ln\frac{4m}{\delta} \rceil$ samples from group i. Then, the refinement $L^{\dagger}_{k+1}$ of L_k that maximizes the estimated improvement potential $\widetilde{\phi}^{+}(L^{\dagger}_{k+1})$ will have error that exceeds the error of the best refinement L^{*}_{k+1} of L_k by at most $\mathrm{err}'^{+}(L^{\dagger}_{k+1}) - \mathrm{err}'^{+}(L^{*}_{k+1}) \leq \epsilon\kappa^2$.*

Proof. Fix any boundary segment T in L^k. Then, we draw $x_{i,1}$, $x_{i,2}, \ldots, x_{i,C_i}$ from the groups $i \in T$. It is not hard to see

$$\begin{aligned}\tilde{s}_T - s_T &= \tfrac{1}{|T|}\sum_{i\in T}\tilde{s}_i - s_i = \tfrac{1}{|T|}\sum_{i\in T}\left(\sum_{j=1}^{C_i}\tfrac{n_i\,x_{i,j}}{C_i}\right) - n_i\mu_i \\ &= \tfrac{1}{|T|}\sum_{i\in T}\left(\sum_{j=1}^{C_i}\tfrac{n_i\,x_{i,j}}{C_i} - \tfrac{n_i\,\mu_i}{C_i}\right)\end{aligned}$$

If $x_{i,j}$ is a sub-Gaussian random variable with parameter σ^2, then $n_i\,x_{i,j}/C_i$ is a sub-Gaussian random variable with parameter $(n_i\,\sigma/C_i)^2$. Using the Hoeffding bound for sub-Gaussian random variables,

$$\begin{aligned}&\Pr\left[|\tilde{s}_T^2 - s_T^2| > \tfrac{\epsilon\kappa^2 m}{6|T|}\right] \\ &= \Pr\left[|\tilde{s}_T - s_T|(\tilde{s}_T + s_T) > \tfrac{\epsilon\kappa^2 m}{6|T|}\right] \\ &\leq \Pr\left[|\sum_{i\in T}\sum_{j=1}^{C_i}\tfrac{n_i}{C_i}(x_{i,j} - \mu_i)| > \tfrac{\epsilon\kappa^2 m}{12a\sum_{i\in T} n_i}\right] \\ &\leq 2\exp\left(-\frac{\epsilon^2\kappa^4 m^2}{288a^2\sigma^2\left(\sum_{i\in T} n_i\right)^2\left(\sum_{i\in T} n_i^2/C_i\right)}\right) \\ &\leq 2\exp\left(-\frac{\epsilon^2\kappa^2 m^2}{288a^2\sigma^2|T|^2\left(\sum_{i\in T} n_i^2/C_i\right)}\right) \\ &\leq \tfrac{\delta}{4m}.\end{aligned}$$

where the last inequality is true because

$$\frac{C_i}{n_i^2} = \frac{288a^2\sigma^2 m}{\epsilon^2\kappa^2}\ln\frac{4m}{\delta} \geq \frac{288a^2\sigma^2|T|^3}{\epsilon^2\kappa^2 m^2}\ln\frac{4m}{\delta}.$$

Therefore, for every boundary segment T of the k-segment approximation L_k, we obtain an estimate $\tilde{s}_T$ of the mean s_T that satisfies $\left|\tilde{s}_T^2 - s_T^2\right| \leq \frac{\epsilon\kappa^2 m}{6|T|}$. □

By replacing C with C_q in line 6 of Algorithm 1 and substituting the calculations for mean of groups and boundary segments to their respective sums, we obtain our incrementally improving visualization generation algorithm for SUM with known group sizes case. Note that here we have $\epsilon\kappa^2$ instead of ϵ: while at first glance this may seem like an amplification of the error, it is not so: first, the scales are different—and visualizing the SUM is like visualizing AVG multiplied by κ—an error by one "pixel" for the former would equal κ times the error by one "pixel" for the latter but are visually equivalent; second, our error function uses a squared ℓ_2-like distance, explaining the κ^2.

Unknown Group Sizes. For this case, we simply employ the known group size case, once the COUNT estimation is used to first approximate the τ_i with $\gamma = \epsilon/24a$. The task of approximating the SUM aggregate function when we do not know the size of each group can be completed by combining the algorithmic tools

described in earlier sections. Specifically, we can use the approach described in the COUNT section to first approximate the size of each group. We can then modify the Algorithm 1 for approximating the AVG function to show the fractional sum. Since we have $\tilde{s}_i = \tilde{\tau}_i \tilde{\mu}_i \kappa_t$, where κ_t denotes the total number of items: $\sum_{i=1}^{m} n_i$, it suffices to run our Algorithm 1 for visualizing the $\tau_i \mu_i$ and multiply all of them by κ_t^2. Therefore, we state the following theorem:

Theorem 7. *Assume we have* $C_i = \lceil \frac{1152\, a^2 \sigma^2 \tilde{\tau}_i^2}{\epsilon^2 m} \ln \frac{4m}{\delta} \rceil$ *samples from group* i*. Then, the refinement* $L_{k+1}^{\dagger}$ *of* L_k *that maximizes the estimated* $\widetilde{\phi^+}(L_{k+1}^{\dagger})$ *will have error that exceeds the error of the best refinement* L_{k+1}^{*} *of* L_k *by at most* $\mathrm{err}'(L_{k+1}^{\dagger}) - \mathrm{err}'(L_{k+1}^{*}) \leq \epsilon \kappa_t^2$.

Proof. Fix any boundary interval T in L^k. Then, we draw $x_{i,1}, x_{i,2}, \ldots, x_{i,C}$ from the groups $i \in T$. It is not hard to see

$$
\begin{aligned}
|\tilde{s}_T - s_T| &= \tfrac{1}{|T|} \left| \sum_{i \in T} \tilde{s}_i - s_i \right| = \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \tilde{\tau}_i \tilde{\mu}_i - \tau_i \mu_i \right| \\
&= \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \tilde{\tau}_i \tilde{\mu}_i - \tilde{\tau}_i \mu_i + \tilde{\tau}_i \mu_i - \tau_i \mu_i \right| \\
&= \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \tilde{\tau}_i (\tilde{\mu}_i - \mu_i) \right| + \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \mu_i (\tilde{\tau}_i - \tau_i) \right| \\
&\leq \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \sum_{j=1}^{C_i} \left(\tfrac{\tilde{\tau}_i\, x_{i,j}}{C_i} \right) - \tilde{\tau}_i \mu_i \right| + \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \mu_i (\tilde{\tau}_i - \tau_i) \right| \\
&= \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \sum_{j=1}^{C_i} \left(\tfrac{\tilde{\tau}_i\, x_{i,j}}{C_i} - \tfrac{\tilde{\tau}_i\, \mu_i}{C_i} \right) \right| + \tfrac{\kappa_t}{|T|} \left| \sum_{i \in T} \mu_i (\tilde{\tau}_i - \tau_i) \right| .
\end{aligned}
$$

Now, we show that each of the term above are less than $\epsilon\, m\, \kappa_t / (24a|T|)$. If $x_{i,j}$ is a sub-Gaussian random variable with parameter σ^2, then $\tilde{\tau}_i\, x_{i,j}/C_i$ is a sub-Gaussian random variable with parameter $(\tilde{\tau}_i\, \sigma / C_i)^2$. Using the Hoeffding bound for sub-Gaussian random variables,

$$
\begin{aligned}
&\Pr \left[\left| \sum_{i \in T} \sum_{j=1}^{C_i} \left(\tfrac{\tilde{\tau}_i\, x_{i,j}}{C_i} - \tfrac{\tilde{\tau}_i\, \mu_i}{C_i} \right) \right| > \tfrac{\epsilon m}{24a} \right] \\
&\leq 2 \exp \left(- \frac{\epsilon^2 m^2}{1152\, a^2\, \sigma^2 \sum_{i \in T} \tilde{\tau}_i^2 / C_i} \right) \\
&\leq \tfrac{\delta}{4m}.
\end{aligned}
$$

Thus, the above expression is true for all T with probability $1 - \delta/2$. Then, the first term in the previous equation is at most $\epsilon m \kappa_t / (24a|T|)$. Let $\gamma = \epsilon/(24a)$. Using the algorithm explained for estimating τ_i in the COUNT section, with the right set of parameter, one can assume with probability $1 - \delta/2$, $|\tilde{\tau}_i - \tau_i|$ is at most γ. Thus, with probability $1 - \delta$, $\tilde{s}_T - s_T$ is at most $\epsilon \kappa_t / (12a)$. Thus, we have

$$
|\tilde{s}_T^2 - s_T^2| = |\tilde{s}_T - s_T|(\tilde{s}_T + s_T) \leq \frac{\epsilon \kappa_t}{12a} \cdot 2\kappa_t a = \frac{\epsilon \kappa_t^2}{6|T|}
$$

Therefore, for every boundary segment T of the k-segment approximation L_k, we obtain an estimate $\tilde{s}_T$ of the mean s_T that satisfies $\left|\tilde{s}_T^2 - s_T^2\right| \leq \frac{\epsilon \kappa_t^2}{6|T|}$. □

Similar to the known group sizes case, by replacing C with C_q in line 6 of Algorithm 1 and substituting the calculations for mean of groups and boundary segments to their respective sums, we obtain our incrementally improving visualization generation algorithm for SUM with unknown group sizes case.

3.5.3 *Selection Attributes*

Consider the following query:

```
Q_T = SELECT X_a, AVG(Y) FROM R GROUP
      BY X_a ORDER BY X_a WHERE Pred
```

Here, we may have additional predicates on X_a or some other attributes. For instance, we may want to view the average delay of all flights landing in ORD airport on December 22nd. Our algorithms still work even if we have selection predicates on one or more attributes, as long as we have an index on the group-by attribute. The sampling engine's bitmap indexes allow us to retrieve, on demand, tuples that are from any specific group i that satisfy the selection conditions specified, by using appropriate AND and OR operators [33].

4. INCVISAGE SYSTEM ARCHITECTURE

Figure 2 depicts the overall architecture of INCVISAGE. The INCVISAGE client is a web-based front-end that captures user input and renders visualizations produced by the INCVISAGE back-end. The INCVISAGE back-end is composed of three components: (A) a *query parser*, which is responsible for parsing the input query Q_T or Q_H (see Section 2.1), (B) a *view processor*, which executes *ISplit* (see Section 3), and (C) a *sampling engine* which returns the samples requested by the *view processor* at each iteration. As discussed in Section 2.1, INCVISAGE uses a bitmap-based sampling engine to retrieve a random sample of records matching a set of adhoc conditions. The sampling engine is the same as the sampling engine used in IFOCUS [32]. At the end of each iteration, the *view processor* sends the visualizations generated to the front-end encoded in *json*. To reduce the amount of json data transferred, only the changes in the visualization (i.e., the refinements) are communicated to the front-end. The front-end then parses the data and generates the visualization.

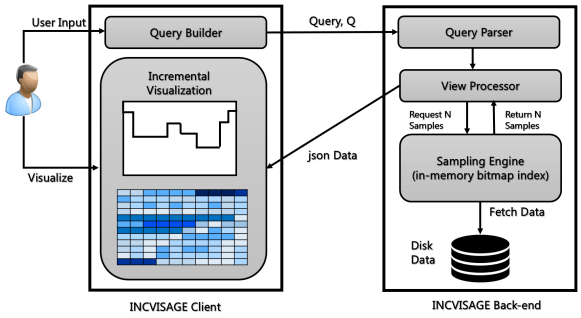


Figure 2: INCVISAGE **Architecture**

The front-end is responsible for capturing user input and rendering visualizations generated by the INCVISAGE back-end. The visualizations (i.e., the segment approximations) generated by the back-end incrementally improve over time, but the users can pause and replay the visualizations on demand. Figure 3 depicts the web front-end for INCVISAGE comprising four parts: (A) a query builder used to formulate queries based on user input; (B) a visualization display pane; (C) a play bar to interact with the incrementally improving visualization being generated by allowing users to pause the visualization generation or rewind to older iterations (i.e., previous segment approximations), and (D) a snapshot pane for saving the image of the current iteration or segment approximation being viewed in case the user wants to compare the current visualization with future ones and identify the differences. There is an additional color legend (E) for heatmaps to allow users to interpret the values of the different heatmap blocks. For the user study described in Section 6, we additionally added a question-answer module (F) to the front-end for submitting answers to the user study questions and also for displaying the points (i.e. the score) obtained by the user.

5. PERFORMANCE EVALUATION

We evaluate our algorithms on three primary metrics: the error of the visualizations, the degree of correlation with the "best" algorithm in choosing the split groups, and the runtime performance.

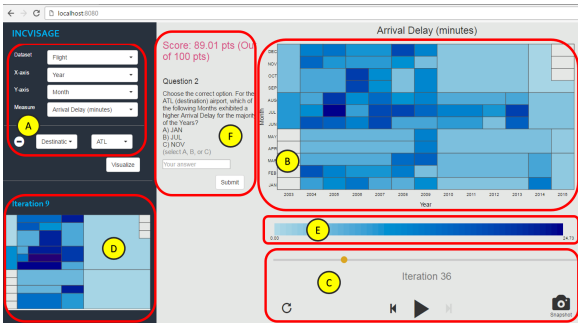


Figure 3: Front End

We also discuss how the choice of the initial samples N_1 and sampling factor α impacts the error and interactivity.

5.1 Experimental Setup

Algorithms Tested. Each of the incrementally improving visualization generation algorithms that we evaluate performs uniform sampling, and take either B (time budget) and f (sampling rate of the sampling engine) as input, or ϵ (desired error threshold) and δ (the probability of correctness) as input, and in either case computes the required N_1 and α. The algorithms are as follows:

ISplit: At each iteration k, the algorithm draws $\frac{N_k}{m}$ samples uniformly from each group, where N_k is the total number of samples requested at iteration k and m is the total number of groups. *ISplit* uses the concept of *improvement potential* (see Section 3) to split an existing segment into two new segments.

RSplit: At each iteration k, the algorithm takes the same samples as *ISplit* but then selects a segment, and a split group within that, all at random to split. Our goal in including this algorithm is to evaluate whether the *improvement potential* based approach can generate visualizations with lower error compared to the random choices.

ISplit-Best: The algorithm simulates the ideal case where the mean of all the groups are known upfront (see Section 3.1). The visualizations generated have the lowest possible error at any given iteration (i.e. for any k-segment approximation) among approaches that perform refinement of previous approximation. We include this algorithm to measure how close the error of *ISplit* is to the lowest possible error when the iterative refinement constraint is respected.

DPSplit: At a high level, at each iteration k, this algorithm takes the same samples as *ISplit*, but instead of performing refinement, *DPSplit* recomputes the best possible k-segment approximation using dynamic programming. We include this algorithm to measure the impact of the iterative refinement constraint. There are two reasons why this algorithm is not preferred: the visualizations change drastically in each iteration, and the dynamic programming computation can be extremely costly online.

DPSplit-Best: This algorithm simulates the case where the means of all the groups are known upfront, and the same approach for producing k-segment approximations used by *DPSplit* is used. The visualizations generated have the lowest possible error among the algorithms mentioned above since they have advance knowledge of the means, and do not need to obey iterative refinement.

Name	Description	#Rows	Size (GB)	E	U
Sensor	Intel Sensor dataset Berkeley Research lab [1].	2.2M	0.73		✓
FL	US Flight dataset [5] .	74M	7.2	✓	✓
T11	2011 NYC taxi trip data for 2011 [3]	170M	6.3	✓	
T12	2012 NYC taxi trip data for 2012 [3]	166M	4.5	✓	
T13	2013 NYC taxi trip data for 2013 [3]	166M	4.3	✓	
WG	Weather data of US cities from 1987–2015 [6]	415M	27	✓	

Table 2: Datasets Used for the Experiments and User Studies. E = Experiments and U = User Studies (Section 6 and 7).

Datasets and Queries. The datasets used in the performance evaluation experiments and the user studies (Section 6 and Section 7) are shown in Table 2 and are marked by ticks (✓) to indicate where they were used. For the US flight dataset we show the results for the attribute *Arrival Delay* (FLA) and *Departure Delay* (FLD). For all three years of the NYC taxi data, we present the results for the attribute *Trip Time*. For the weather dataset, we show results for the attribute *Temperature*. For all the datasets, we remove the outliers. To verify our sub-Gaussian assumption, we generated a Q-Q plot [49] for each of the selected attributes to compare the distributions with Gaussian distributions. The FL, T11, T12, and T13 datasets all exhibit right-skewed Gaussian distributions while WG exhibits a truncated Gaussian distribution We also performed histogram analysis of the datasets to further confirm the observations obtained from Q-Q plot analysis. The results can be found in Appendix 4. Unless stated explicitly, we use the same query in all the experiments—*calculate the average of an attribute at each day of the year*. Here, the number of groups (days), $m = 366$.

Metrics. We use the following metrics to evaluate the algorithms:

Error: We measure the error of the visualizations generated at each iteration k via $\text{err}(L_k)$ (see Section 2.3). The average error across iterations is computed as: $\widetilde{\text{err}}(L_k) = \frac{1}{m}\sum_{k=1}^{m}\text{err}(L_k)$.

Time: We also evaluate the *wall-clock execution time*.

Correlation: We use Spearman's ranked correlation coefficient to measure the degree of correlation between two algorithms in choosing the order of groups as split groups. We use the order in which the groups were selected as split groups by an algorithm to compute a ranked list for the purpose of applying Spearman's correlation coefficient. Spearman's ranked correlation coefficient captures the correlation between two ranked lists. If we consider the iteration at which a group was chosen as a split group as the rank of that group, we can get a ranked list of the groups for each of the algorithms. Then, we can compute the correlation between any two ranked lists. Let $e_1, \cdots, e_m$ and $f_1, \cdots, f_m$ are the two ranked lists where e_i and f_i are the ranks of group i assigned by algorithm E and F, respectively. The Spearman's ranked correlation coefficient (r) is defined as follows:

$$r(E,F) = 1 - \frac{6\sum_{i=1}^{m}(e_i - f_i)^2}{m(m^2-1)}$$

A value close to 1 (-1) for the Spearman's coefficient indicates positive (negative) correlation between the two lists, while a value close to 0 indicates no correlation.

Interactivity: We use a new metric called interactivity (defined in Section 2.3) to select appropriate parameters for *ISplit*. Interactivity is essentially the average waiting time for generating the segment approximations. We explore the measure in Section 5.5.

Implementation Setup. We evaluate the runtime performance of all our algorithms on a bitmap-based sampling engine [33]. In addition, we implement a *SCAN* algorithm which performs a sequential scan of the entire dataset. This is the approach a traditional database system would use. Since both *ISplit* and *SCAN* are implemented on the same engine, we can make direct comparisons of execution times between the two algorithms. All our experiments are single threaded and are conducted on a HP-Z230-SFF workstation with an Intel Xeon E3-1240 CPU and 16GB memory running Ubuntu 12.04 LTS. We set the disk read block size to 256KB. To avoid any speedups resulting from the file buffer cache, we perform all the I/O operations using Direct I/O. Unless explicitly stated we assume the time budget $B = 500ms$ and use the parameter values of $N_1 = 25000$, $\alpha = 1.02$ (with a geometric decrease), and $\delta = 0.05$. The choice of the parameters is further explored in Section 5.5. All experiments were averaged over 30 trials.

5.2 Comparing Execution Time

In this section, we compare the *Wall Clock* time of *ISplit*, *DPSplit* and *SCAN* for the datasets mentioned in Table 2.

Summary: ISplit is several orders of magnitude faster than *SCAN* in revealing incremental visualizations. The completion time of *DPSplit* exceeds the completion time of even *SCAN*. Moreover, when generating the early segment approximations, *DPSplit* always exhibits higher latency compared to *ISplit*.

We depict the wall-clock time in Figure 4 for three different datasets for *ISplit* and *DPSplit* at iteration 10, 50, and 100, and at completion, and for *SCAN*. First note that as the size of the dataset increases, the time for *SCAN* drastically increases since the amount of data that needs to be scanned increases. On the other hand, the time for completion for *ISplit* stays stable, and much smaller than *SCAN* for all datasets: on the largest dataset, the completion time is almost $\frac{1}{6}^{th}$ that of *SCAN*. When considering earlier iterations, *ISplit* performs even better, revealing the first 10, 50, and 100 segment approximations within ≈ 5 seconds, ≈ 13 seconds, and ≈ 22 seconds, respectively, for all datasets, allowing users to get insights early by taking a small number of samples—beyond iteration 50 the refinements are minor, and as a result, users can terminate the visualization early if need be. Compared to *SCAN*, the speed-up in revealing the first 10 features of the visualization is $\approx 12X$, $\approx 22X$, and $\approx 46X$ for the FL, T11 and WG datasets. Lastly we note that *ISplit* reveals each increment within the 500ms bound for interactivity, as required.

When comparing *ISplit* to *DPSplit*, we first note that *DPSplit* is taking the same samples as *ISplit*, so its differences in execution time are primarily due to computation time. We see some strange behavior in that while *DPSplit* is worse than *ISplit* for the early iterations, for completion it is much worse, and in fact worse than *SCAN*. The dynamic programming computation complexity depends on the number of segments. Therefore, the computation starts occupying a larger fraction of the overall wall-clock time for the latter iterations, rendering *DPSplit* impractical for latter iterations. These observations are confirmed in Figure 5—for *DPSplit*, the CPU time accounts for the major portion of the wall clock time. As the number of iterations increases, the CPU time increases drastically. By the time *DPSplit* completes, the CPU time exceeds the wall clock time of *SCAN*. Even for earlier iterations, *DPSplit* is worse than *ISplit*, revealing the first 10, 50, and 100 features within ≈ 7 seconds, ≈ 27 seconds, and ≈ 75 seconds, respectively, as opposed to 5, 13, and 22 for *ISplit*. Thus, at the same time as *ISplit* has completed 100 iterations, *DPSplit* has only completed 50.As we will see later, this additional time does not come with a commensurate decrease in error, making *DPSplit* much less desirable than *ISplit* as an incrementally improving visualization generation algorithm.

5.3 Incremental Improvement Evaluation

We now compare the error for *ISplit* with *RSplit* and *ISplit-Best*.

Summary: (a) The error of *ISplit*, *RSplit*, and *ISplit-Best* reduce across iterations. At each iteration, *ISplit* exhibits lower error in generating visualizations than *RSplit*. *(b) ISplit* exhibits higher correlation with *ISplit-Best* in the choice of split groups, with ≥ 0.9 for any N_1 greater than 25000. *RSplit* has near-zero correlation overall.

Figure 6 depicts the iterations on the x-axis and the ℓ_2-error on the y axis of the corresponding segment approximations for each of the algorithms for two datasets (others are similar). For example, in Figure 6, at the first iteration, all three algorithms obtain the 1-segment approximation with roughly similar error; and as the number of iterations increase, the error continues to decrease. The error for *ISplit* is lower than *RSplit* throughout, for all datasets, justifying our choice of *improvement potential* as a good metric to guide splitting criteria. *ISplit-Best* has lower error than the other two, this is because *ISplit-Best* has access to the means for each group beforehand. For the WG dataset, *ISplit-Best* and *ISplit* perform roughly similarly; this is because there is less skew in that dataset amongst the samples taken; and the errors are low because the trendline ends up having a single prominent feature—a bell-shape.

5.4 Releasing the Refinement Restriction

Next, we compare *ISplit* with *DPSplit* and *DPSplit-Best*.

Summary: Given the same set of parameters, *DPSplit* and *ISplit* have roughly similar error; as expected *DPSplit-Best* has much lower error than both *ISplit* and *DPSplit*.

Figure 7 depicts the error across iterations for *ISplit* (our best online sampling algorithm from the previous section), *DPSplit*, and *DPSplit-Best* for all the datasets—we aim to evaluate the impact of the refinement restriction, and whether it leads to substantially lower error. From Figure 7, at each iteration *DPSplit-Best* has the lowest error, while *ISplit* and *DPSplit* have very similar error. Thus, in order to reduce the drastic variation of the segment approximations, while not having a significant impact on error, *ISplit* is a better choice than *DPSplit*. Furthermore from Section 5.2, we found that *ISplit*'s execution time is much more reasonable than *DPSplit*. Once again, for WG, the errors are all very similar due to the single prominent feature in the visualization. Note also that we noticed cases, especially when N_1 is small, where *DPSplit* is outperformed by *ISplit* in the earlier iterations, since it may "overfit" based on a few skewed samples.

5.5 Optimizing for Error and Interactivity

The goal of an incrementally improving visualization generation algorithm is to provide an approximation of the original visualization at interactive speed. On the other hand, generating highly inaccurate approximations is also not desirable. Hence, the algorithms need to optimize for both accuracy (error) and interactivity. So far, we have kept the sampling parameters fixed across algorithms; here we study the impact of these parameters. Specifically, we evaluate the impact of the initial sample (N_1) and sampling factor (α) on the error-interactivity trade-off. We also show that our simulations resulting from theoretical claims in Section 3.4 match the experimental results.

Summary: We find: *(a)* Geometrically decreasing the sample complexity per iteration leads to higher interactivity. *(b)* Given the time budget (B), only a small set of sampling factors (α) improves interactivity. *(c)* The theoretical and experimental error-interactivity trade-off curves behave essentially the same, producing similarly shaped knee regions.

5.5.1 Parameter Selection

We now empirically evaluate the trade-off between error and interactivity. We focus on "decreasing" sampling factors, i.e., those that result in the sample complexity decreasing across iterations—we have found that "increasing" sampling factors lead to significantly worse λ (i.e., interactivity), while error is not reduced by much. We consider both geometric and linear decrease, as well as upfront sampling (Section 3.4). Figure 8 captures the trade-off between average error (across iterations) on the y axis and logarithm of interactivity, i.e., $\log \lambda$ on the x axis for the Flight (FLA), T11 and WG dataset (other datasets are similar)—focus on the circles and triangles for now. Each line in the chart corresponds a certain number of initial samples N_1, and either geometric decrease (denoted with a "/"), or a linear one (denoted with a "-"). Each point in each line corresponds to a specific value of α. For all lines, we find a similar behavior or shape as α increases, which we explain for $N_1 = 25000$ for geometric and linear decrease, depicted again in Figure 8d with α annotated. If we follow the geometric decrease points (circles) Figure 8b, we start at $I \approx 6.75$ at the point corresponding to $\alpha = 1$ for geometric decrease and 0 for linear decrease, and then as α is increased the points move to the left—indicating that the interactivity improves, while error increases slightly. Then, we have a *knee* in the curve—for any $\alpha > 1.028$, the trails start

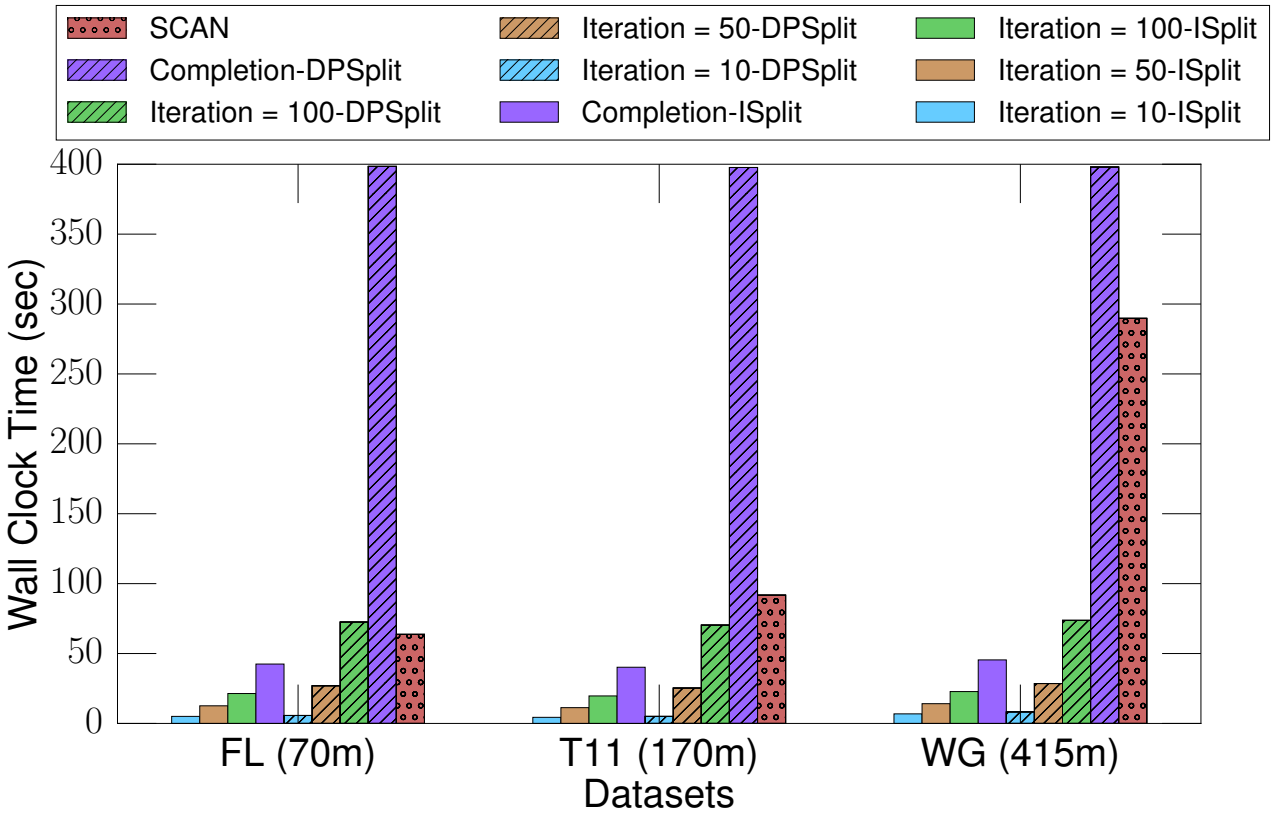


Figure 4: Comparison of Wall Clock Time.

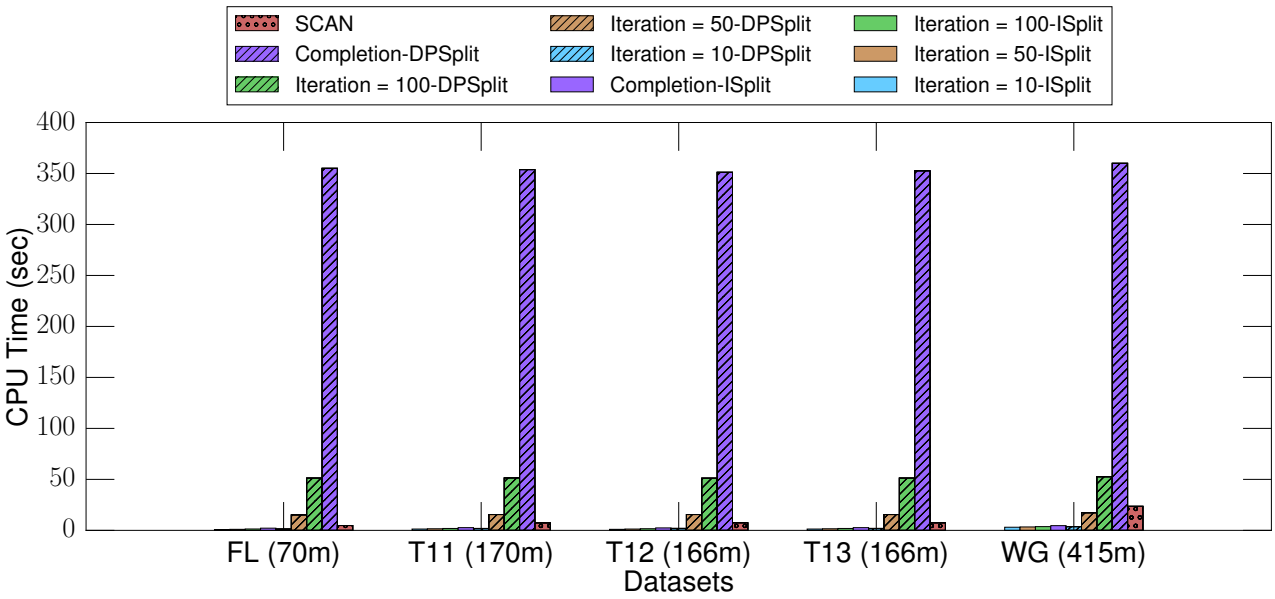


Figure 5: Comparison of CPU Time.

moving back to the right (lower interactivity) but also see a simultaneous increase in error. A similar knee is seen if we trace the linear decrease points (triangles)—note that $\alpha = 1$ is shared between the linear and geometric decrease—here, the sampling is constant across iterations. For other values of α depicted for the linear decrease points, this indicates the reduction in the number of samples per round, e.g., 50, 500, 1000. This behavior of a knee in the curve is seen for all N_1 values. We also find that for the same N_1, the linear decrease has worse interactivity compared to geometric decrease. Finally, on examining the upfront sampling

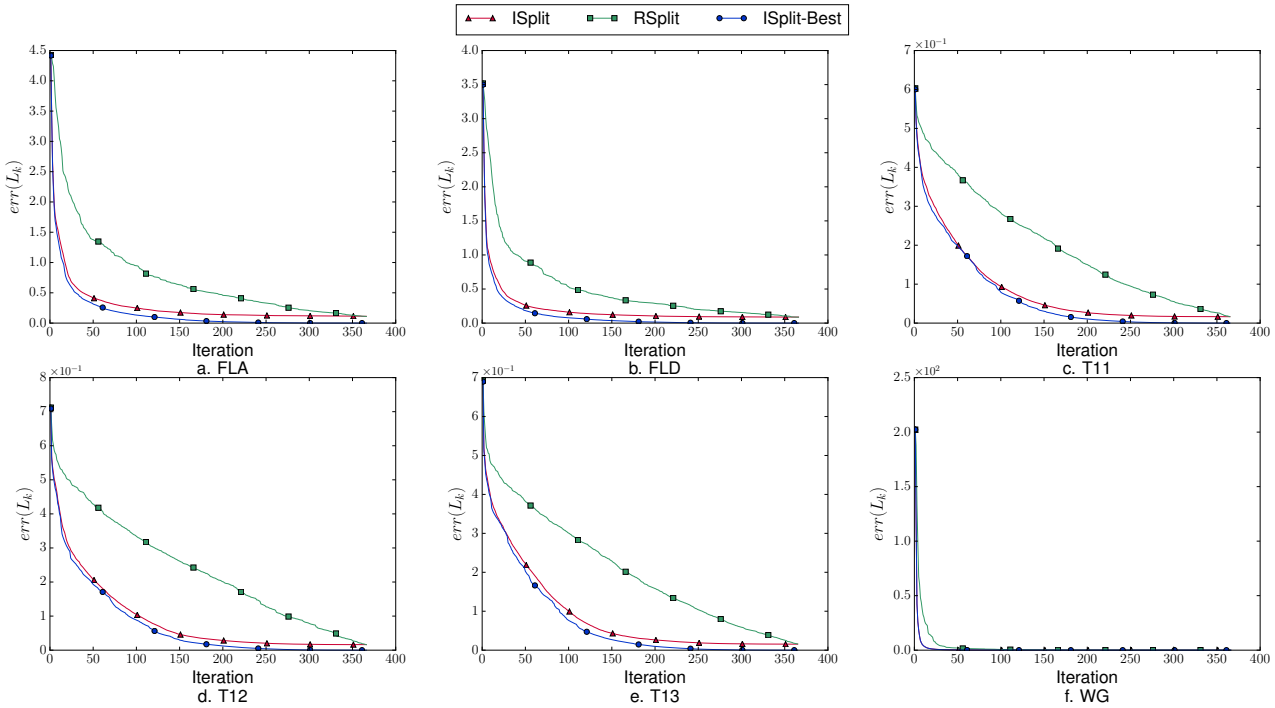


Figure 6: Comparison of the ℓ_2 squared error of *ISplit*, *Rsplit*, and *ISplit-Best*.

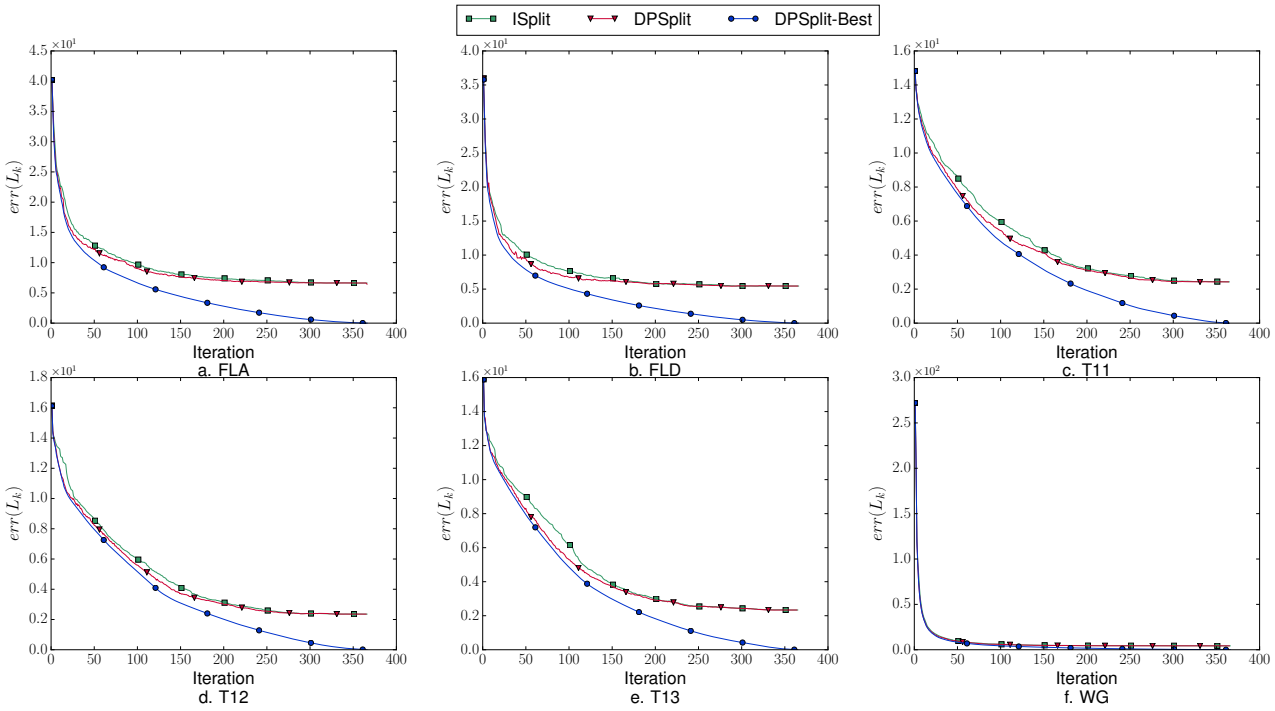


Figure 7: Comparing the ℓ_2 squared error of the *ISplit*, *DPSplit* and *DPSplit-Best*

scheme (squares), we find that both geometric decrease and linear decrease have much better interactivity and lower error.

Overall, when selecting parameters, we would like to identify parameters that result in the favorable knee region of the curve. We find that $\alpha \in [1.0, 1.028]$, with N_1 relatively small helps us stay in this region empirically. We select $\alpha = 1.02$ to balance between error and interactivity if we set the maximum allowable delay per iteration $B = 500ms$ [38]. Based on our sampling rate, fetching 25000 samples takes around 500ms; thus we set $N_1 = 25000$. From our theoretical results in Section 3.4, the range of α for this

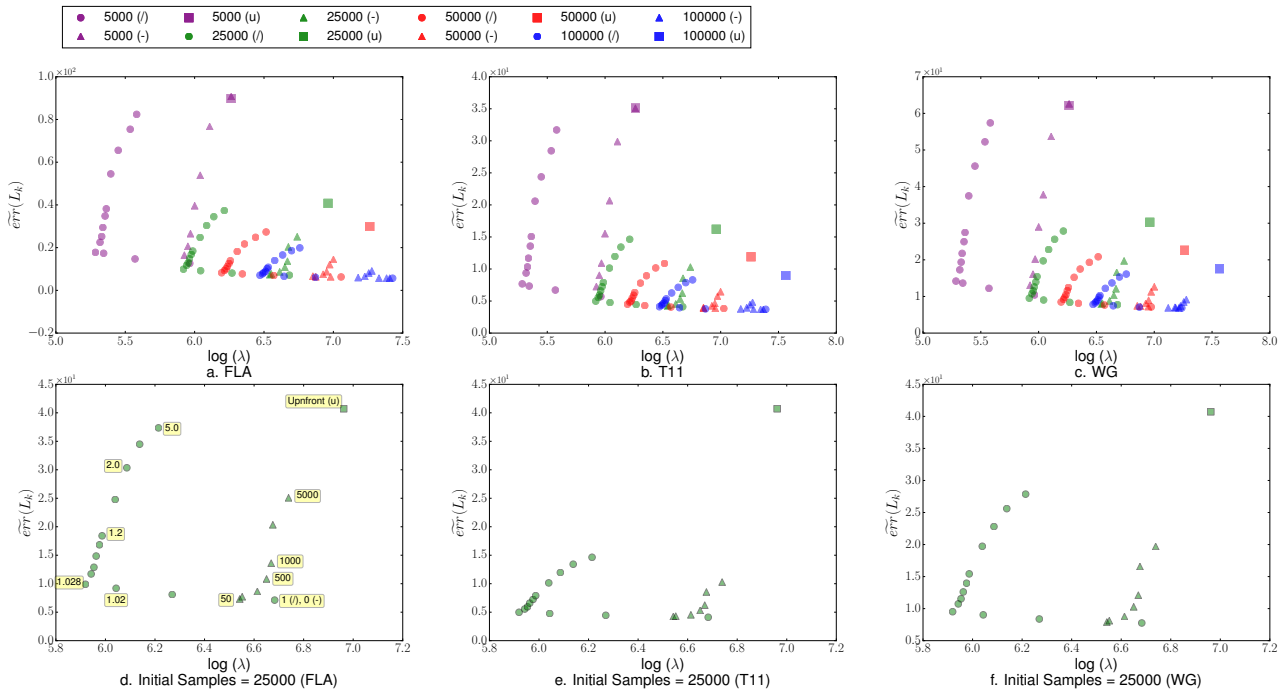


Figure 8: Error-interactivity trade off curve. (/) = Geometric decrease, (-) = Linear decrease, (u) = Upfront Sampling.

N_1 was $[1, 1.028]$, so our experiments agree with the theory. Next, we simulate the error-interactivity trade-off curve for the sampling approaches using the expressions of error and interactivity obtained in Section 3.4.

5.5.2 *Simulations vs. Empirical Observations*

Figure 10 captures the trade-off between the upper bound of the error averaged (across iterations) on the y axis and logarithm of interactivity, i.e., $\log \lambda$ on the x axis for the Flight (FLA), T11 and WG dataset.

Each line in the chart corresponds a certain number of initial samples N_1, and either geometric decrease (denoted with a "/" and represented by circles), or a linear one (denoted with a "-" and represented by triangles). Furthermore, for each N_1, we also plot the Error and interactivity pair for upfront sampling (denoted by "u" and represented by squares). For geometric decrease, for all lines, we find a similar behavior as our empirical results—a knee shape emerges as α increases starting from 1. The theoretical value for the optimal decrease factor α^* is annotated for each initial sample N_1. Furthermore, for each N_1 the optimal α^* is highlighted by a red arc and is the point with the best interactivity in each line—same as the empirical observation. For the linear decrease, given the same decrease factors β used in the experiments, the simulation results match the experimental results. Similar to the empirical results, upfront sampling has the worst error and interactivity than all the other approaches. Therefore, we can clearly see that the simulation results mimic the empirical results obtained in Section 3.4.

5.5.3 *Error vs. Sample Complexity*

Figure 9 captures the correlation between *ISplit-Best* and both of *ISplit*, and *RSplit* in terms of the choice of split groups. We run several simulations of both *ISplit*, and *RSplit* with different initial samples (N_1) while setting $\alpha = 1.02$. Therefore, as N_1 increases, the overall sample complexity of the simulation also increases. The x-axis represents the initial samples of the simulations while the y-axis represents the Spearman's coefficient ($r(E, F)$) of the corresponding simulation. For a fixed α, higher the number of initial samples, higher the overall sample complexity of the algorithm. Figure 9 confirms that as the sampling complexity increases, *ISplit* starts to exhibit higher correlation with *ISplit-Best* while choosing the split groups. Beyond a certain sampling complexity $r(E, F)$ starts to taper-off—indicating that further increasing the sampling complexity will not improve the correlation. *RSplit* on the other hand, is completely uncorrelated to *ISplit-Best*. For small sampling complexity ($N_1 = 500$, the first green circle in the plots) even *ISplit* does not exhibit any correlation with *ISplit-Best*. Due to insufficient sampling, the choice of split groups are so erroneous that it seems as if *ISplit* is choosing split groups randomly rather than intelligently. For our choice of initial samples $N_1 = 25000$ (the circle highlighted in red), *ISPlit* exhibits high correlation ($r(E, F) > 0.78$) to *ISplit-Best* for all three datasets.

6. EVALUATION: INTERPRETABILITY

We now present an evaluation of the usability of INCVISAGE, and more broadly evaluate how users interpret and use incrementally improving visualizations. We aim to address the following questions: 1) Are users willing to use *approximate* visualizations if it saves them time? 2) How confident are users when interpreting visualizations? 3) Of the two types of visualizations, do users prefer the heatmap or the trendline visualization?

6.1 Study Design and Participants

The study consisted of five phases: (a) an introduction phase that explained the essential components of INCVISAGE, (b) an exploration phase that allowed the participants to familiarize themselves with the interface by exploring a sensor dataset (see Section 5.1), (c) a quiz phase where the participants used the same interface to answer targeted questions on the flight dataset, (d) an interview to collect qualitative data during the quiz phase, and (e) a closing survey to obtain feedback on INCVISAGE. We describe the quiz phase in Section 6.2. The interview and survey phases are presented in Section 6.3. All of the studies were conducted by in the same lab

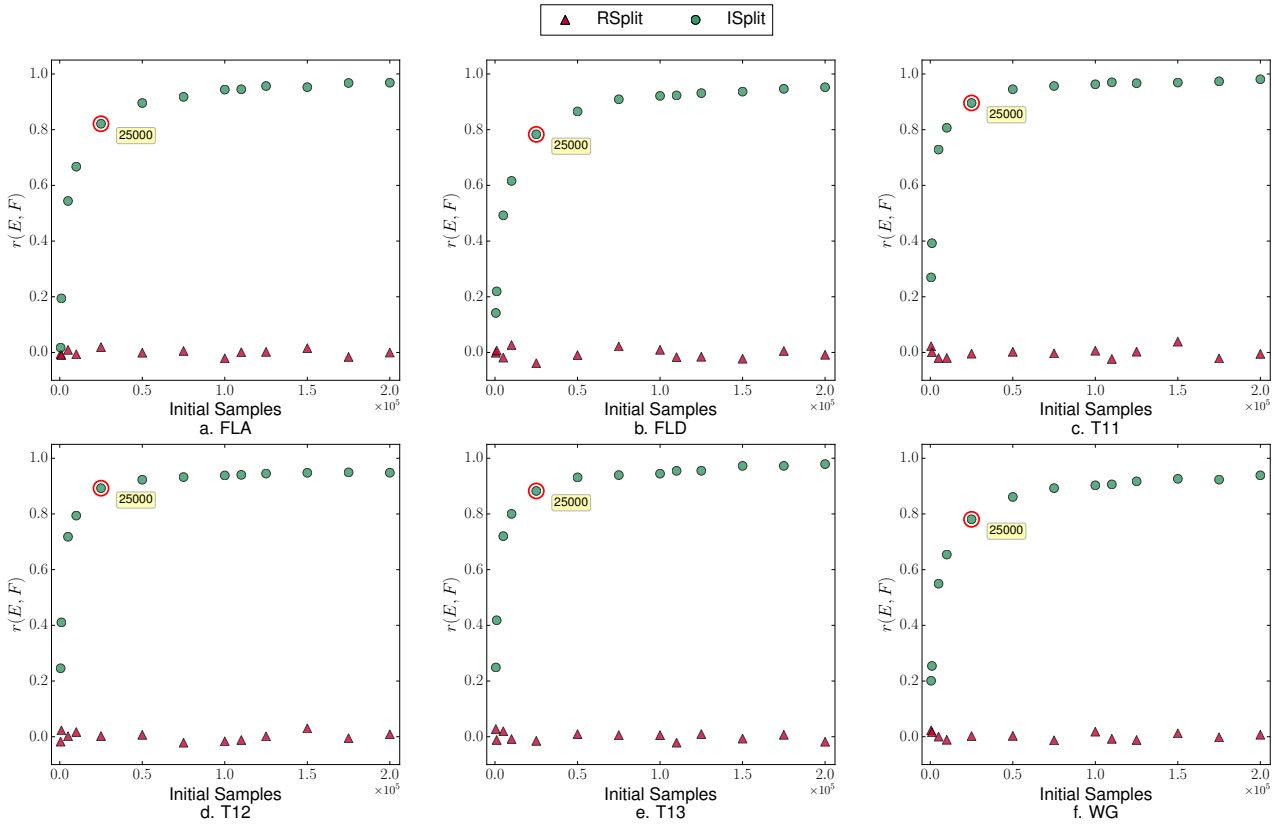


Figure 9: Spearman's Ranked Correlation Coefficient with varying sample complexity. E = {*ISplit*, *RSplit*} and F = *ISplit-Best*.

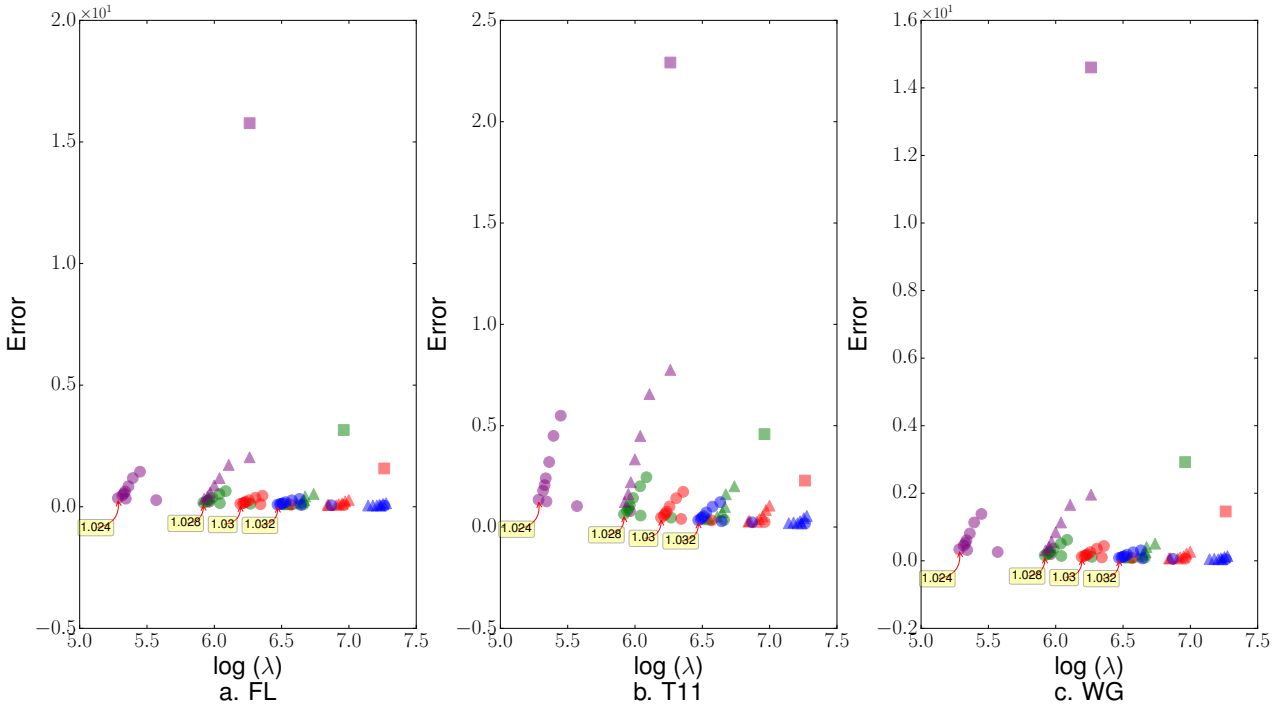


Figure 10: Simulation of the error-interactivity trade off curve.

setting with the same two researchers. The first researcher led the introduction and exploration phases; the second led the quiz, interview, and survey phases. Our study was conducted prior to the development of *ISplit*, and was meant to assess the utility of incremental visualizations—nevertheless, we ensured that the interactivity criteria of 500ms per iteration was met [38].

We recruited 20 participants (11 male, 9 female) via flyers across a university and via a university email newsletter. Our participants included 11 graduate students (8 in computer science), one business undergraduate student, two researchers, and six university employees. All of the participants had experience with data analysis tools such as R, Matlab, and/or Excel. The median age of the participants was 28 (μ = 32.06; σ = 11.9). Out of the 20 participants, 7 reported that they analyzed or worked with data "Daily", 7 answered "Monthly", while the remaining participants answered "Weekly". The duration of the sessions ranged from approximately 60 to 90 minutes. Each participant received $10 per hour at the end of their session.

6.2 The Quiz

We now explain the design of and findings from the quiz phase.

6.2.1 The Quiz Phase Design

The purpose of the quiz phase was to evaluate whether participants were willing to compromise accuracy to make rapid decisions when posed various types of questions. One way to capture such behavior is via a point based system where early submissions are rewarded while late submissions are penalized. With this incentive, participants would be encouraged to submit their answers quickly. We describe the scoring function we used in our point based system later in this section. We first categorize the questions used during the quiz phase.

We used two types of quiz questions: extrema-based (E1-7), and range-based (R1-7). These questions are listed in Table 3. The extrema-based questions asked a participant to find the highest or lowest values in a visualization. The range-based questions asked a participant to estimate the average value over a time range (e.g., months, days of the week). The purpose of such a categorization is to evaluate the accuracy and confidence of participants in finding both specific points of interest (extrema) and patterns over a range (range) when given INCVISAGE. The extrema based questions were free form questions; the range based questions were multiple choice questions. Two of the range based questions were formatted as free form questions but, operationally, were multiple-choice questions with seven possible options (e.g., the day of the week). To prevent order effects, ten participants started the quiz with heatmaps; the other ten started with trendlines. Additionally, we randomized the order of the questions for each participant. Next, we define the scoring function used in assessing quiz performances.

The Scoring Function. The scoring function relied on two variables: the iteration number at which the participant submitted an answer—the higher the iteration the lower the score, and whether or not that answer was accurate—the higher the accuracy the higher the score. The participants were informed prior to the quiz phase that the score was based on these two variables, but the exact function was not provided. The maximum attainable score for each answer was 100 points.The score was computed as a product $S = P \cdot A$, where P was based on the iteration, and A on the accuracy. If a participant submitted an answer at iteration k, we set $P = \frac{m-k}{m}$, i.e., the fraction of the remaining number of iterations over the total number of iterations, m. Thus, based on the definition of P, any answer submission at the last iteration receives zero points, irrespective of question type. To compute A, let c be the correct answer to a question, and let u be the answer submitted by the participant. The accuracy A of a multiple choice question is 0 if $u = c$ and 1 otherwise. The accuracy A of a free-form question is $1 - \frac{|u-c|}{|c|}$, measuring how far the answer is from the correct one. For the free form questions, submitting an incorrect answer that is close to the actual answer could result in a high score. Since the free form questions were range based, participants could submit an approximate answer early to gain more points. On the other hand, for the multiple choice questions, a correct answer submitted at later iterations would yield lower scores.

Interface Issues. Analyzing the quiz results proved more difficult than expected. Out of 280 total submissions, 10 submissions had interface issues—4 of those occurred due to ambiguity in the questions (*R1, R4, R7* in Table 3), while others were due to mistakes made by the participants in selecting the visualization to be generated.For example, for one question (*R1*), dealing with *departure delay* two participants selected the "departure airport" attribute as opposed to "origin airport". A similar issue arose with questions *R4* and *R7*. The ambiguity arose from attribute names in the original dataset—to maintain consistency, instead of renaming these attributes on-the-fly to fix the ambiguity, we did not make any changes. We explicitly separate out interface issues when analyzing the results.

6.2.2 Quantitative Analysis

In discussing the participants' quiz performance, we first investigate their answer submission trends. Finally, we report the progress of each participant. As the participants interacted with the tool, we recorded their responses to each question, the iterations at which a participant submitted an answer, the time taken to submit an answer after starting the visualization, and the points obtained based on the scoring function.

Summary: The majority of the submissions for both trendlines (75%) and heatmaps (77.5%) were within the first 25% of the total number of iterations. Even though the participants chose to trade accuracy for time in both cases, they did so with reasonably high accuracy.

Trading accuracy for time. First, we analyze how the participants traded accuracy for time. Figure 11 shows a dot graph analysis of the participants' submission trends as a function of iteration. For both the trendline and heatmap visualizations, we separated the statistics for the correct and incorrect submissions (Figure 11a and 11b). Correct submissions are represented by *green* circles. Incorrect submissions are either represented by *blue* circles (interface issue) or *red* circles. The x-axis represents at what fraction of the total number of iterations an answer was submitted.

For trendlines, the majority of the submissions (75%) were made at around 25% of the total iterations, except for question *E4* (Figure 11a). Question *E4* asks for a *day of the year* that shows the highest *departure delay* across all years. During the study, we discovered that the departure delays for December 22 and December 23 were very similar. Due to the proximity of the values, these dates were not split into separate groups until a later iteration. One participant even waited until iteration 237 (out of 366 iterations) to submit their answer. Figure 11b shows the trends for heatmaps. Similar to trendlines, the majority of the participants (77.5%) chose to submit answers earlier (within 25% of the iterations) except for questions *R5* and *R7*, where once again there were two *days of the week* with very similar delays, leading to the relevant heatmap block being split in a later iteration.

The submission trends indicate that participants opted for a higher reward and submitted their answers as early as possible, when the visualizations became stable or when obvious extrema emerged, trading accuracy for time. This observation is confirmed in Section 6.3. Figure 12 plots the accuracy of all the submissions according to the scoring functions described in Section 6.2.1. The x-axis represents at what fraction of the total number of iterations an answer was submitted; *accuracy* appears on the y-axis. For both the trendline (Figure 12a) and heatmap (Figure 12b) visualizations, the accuracy of the majority of the submissions is reasonably high.

Submission trends. Here we show at what fraction of the iterations (% iteration) the participants typically opted to submit their answers. We also analyze the submission trends for the different

Type	Extrema-based questions	Range-based questions
Trendline	**E1.** In the state of NY (destination), which day of the year suffers most from the delay caused by the *National Aviation System (NAS delay)*?	**R1.** Choose the correct option based on the day of the year. Anyone traveling from LGA airport (origin) during _____ has to suffer from the highest *Departure Delay*. A) Early January B) Summer C) Late December
	E2. Find the busiest day of the year in the ORD (origin) airport. The higher the *Taxi Out* time, the busier the airport.	**R2.** Choose the correct option based on the day of the year. UA (carrier) aircrafts have the worst *Aircraft Delay* statistics on: A) Jan 01- Jan 10 B) Jun 10- Jun 20 C) Dec 01- Dec 10
	E3. Find the Month that has the day (of month) with the shortest *Arrival Delay*.	**R3.** Choose the correct option. Overall, the *Arrival Delay* is the worst in: A) First half of January B) Mid July C) Last half of December
	E4. Which Day of the year has the highest *Departure Delay*?	
Heatmap	**E5.** For the *Week of Year* Vs.*Year* heatmap, which week of the year had the highest *Departure Delay*?	**R4.** Choose the correct option. Which of the following months has the most *Days of Month* with a low *Arrival Delay*? A) Feb B) Jul C) Oct
	E6. Find the (day of month, month) pair that had the highest *Aircraft Delay* in DEN (destination) airport.	**R5.** Over the course of months, which *Day of Week* exhibits a higher *Carrier Delay* for AA (Carrier)?
	E7. Find the (day of month, month) pair in the year 2013, that had the highest *Weather Delay*.	**R6.** Choose the correct option. For the ATL (destination) airport, which of the following months exhibited a higher *Arrival Delay* for the majority of the years? A) Jan B) Jul C) Nov
		R7. Over the Years, which *Day of Week* had the highest *Arrival Delay* in the city of Atlanta- GA (departure city)?

Table 3: Categorization of the user study questions.

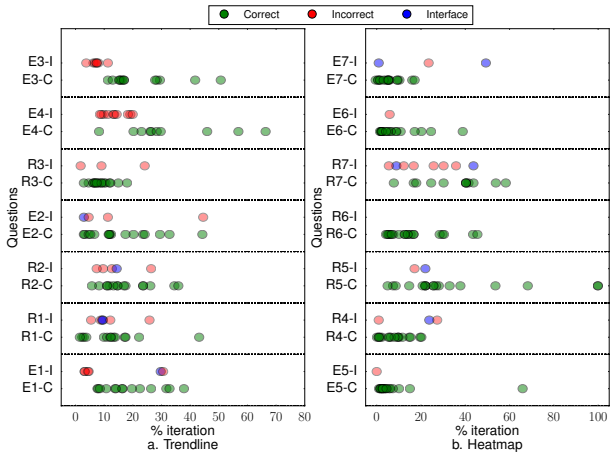


Figure 11: Per-question statistics for the iterations at which participants submitted answers for trendlines (l) and heatmaps (r).

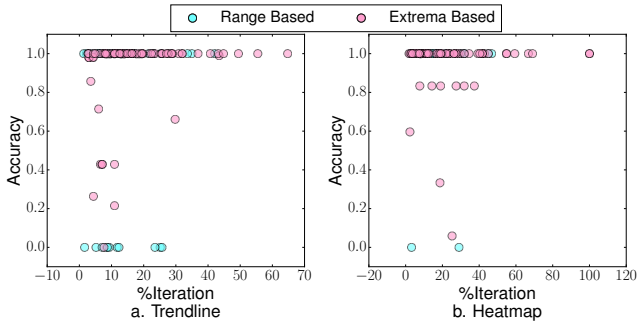


Figure 12: Accuracy vs Submission (% iteration) statistics for (a) trendline and (b) heatmap visualizations.

question types in each visualization. Figure 13 presents the box and whisker plot of the answer submission trends. For trendlines (Figure 13a), the range-based questions were submitted earlier (75% of the submissions at % iteration $\approx$ 15%) compared to the extrema-based questions (75% of the submissions at %iteration $\approx$ 28%). This difference in submission trends across types may be due to the fact that the range based questions asked participants to find more coarse grained information (e.g., the delay statistics in a specific range) which may be easier than finding peaks and valleys. Also the range-based questions were multiple choice—the hints provided by the multiple choice options may have provided guidance to the participants. We see the opposite trend for heatmaps (Figure 13b); the extrema-based questions were submitted earlier compared to the range-based questions. Comparison of the submission trends of the extrema-based questions for trendlines and heatmaps shows that heatmap submissions occurred much earlier. This may be due to the fact that the color dimension of the heatmap helped users compare contrasting blocks, while the continuous generation of peaks and valleys in the trendline led to abrupt changes in the visualization that were difficult to interpret. Hence, participants waited a bit longer for the trendline visualization to stabilize, whereas for the heatmap, the changes in color were not as abrupt and the participants were more confident when they spotted an extremum.

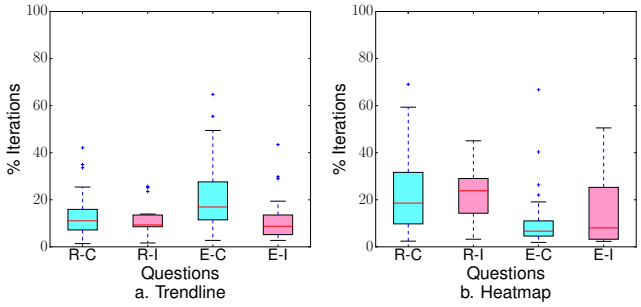


Figure 13: Per category statistics of iterations at which participants submitted answers for (a) trendline and (b) heatmap. Extrema = E, Range = R, Correct = C, Incorrect = I.

Analyzing the Participants. In this section, we analyze the progress of each participant individually. Figure 14 shows a pivot table with graphical marks that depict the progress of the participants during the quiz phase. Each row in the table corresponds to one participant. Each cell in a row, with the exception of the last two, corresponds to the questions that participant answered. Although all participants answered the same set of questions, the order varied due to randomization. The last two columns show the average points obtained by the participant and the average of the percentage of iterations at the point the participant to submitted their answer. Both quantities are represented by circles. The higher the number of points, the larger the radius of the circle – while the lower the iteration percentage, the larger the radius of the circle. For ease of analysis, we divide both the points and the iteration percentages into five ranges. The point ranges are: $\leq$ 55, 56-65,66-75,76-85 and $>$ 85 while the iteration percentage ranges are: $\leq$ 10%, 11-15%, 16-20%, 21-25% and >25%. All points falling into the same range are represented by a circle with the same radius. Similarly, all the iteration percentages falling in the same ranges have the same radius. The participants whose last two cells

contain larger circles performed better during the study.
The first 12 rows are participants with a computer science (CS)

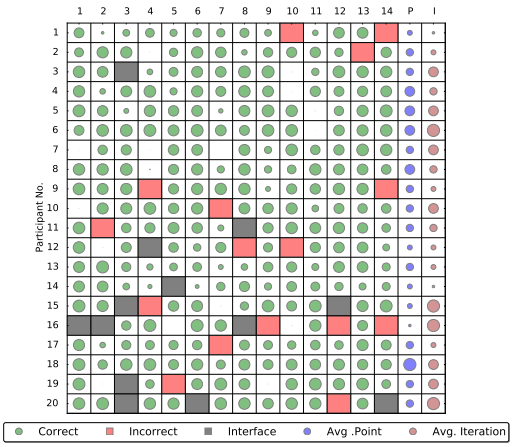


Figure 14: Analyzing the participants.

background, while the remaining participants did not have a CS background. Aside from observing that many participants answered their final three to four questions correctly, participants exhibited no patterns during the quiz. The median average point and average iteration percentage were 70.73 and 14.77%, respectively. Only one participant, P18 answered all of the questions correctly with an average iteration percentage of 13.37% and with the highest point average (86.62). P6 submitted their answers faster than all other participants with a 7.06 iteration percentage, while attaining the second highest point average (84.73). P16 suffered most from interface related issues obtained the lowest point average (38.97).

6.3 Interview and Survey Phase

We now present a qualitative analysis of the participants' perceptions based on their interview and survey responses.

Summary: Participants were highly confident (confidence = 8.5 out of 10) when they submitted answers for both visualization types. Some participants, however, suggested providing explicit guarantees to further boost their confidence. Both trendline and heatmap visualizations were highly interpretable.

Interview. We conducted semi-structured interviews to gauge our participants' motivations for stopping the visualization at a certain iteration, and their confidence of their answers. The main motivations for terminating a visualization were the emergence of obvious extrema ($N = 5$), gaining sufficient confidence in an answer ($N = 10$), or both ($N = 5$). When asked to rate their confidence at the time of submission on a scale of 1 to 10 (10 being the most confident), most participants rated their confidence very high (μ = 8.5 and σ = 1.03 out of 10). However, some participants ($N = 4$) indicated that they would have continued until the final iteration if they were freely exploring the data (and not in an assessment setting). If they were pressed for time or the dataset was very large, they would choose to stop before the final visualization iteration, but likely at a later iteration than the iteration they stopped at in our quiz phase. One of those participants (P8) mentioned that providing an explicit guarantee along with the visualizations would further increase the confidence level when making decisions.

Survey. The survey consisted of seven Likert scale questions to measure the interpretability of the visualizations and the usability of INCVISAGE. Also, there were three free-form questions to collect the participants' feedback on INCVISAGE. The heatmap and trendline visualizations received similar average ratings (out of 5) for interpretability (heatmap: μ = 4.45; σ = 0.51; trendline: μ = 4.50; σ = 0.83) and satisfaction levels (heatmap: μ = 4.55; σ = 0.60; trendline: μ = 4.50; σ = 0.83).

This was not because all participants liked the heatmap and the trendline equally, but rather because the number of participants who preferred each visualization was evenly divided: seven participants preferred the heatmap, six preferred the trendline, and seven rated both visualizations equally. While some participants found each visualization useful for different purposes, some were more enthusiastic about their preference for one type of visualization over the other. Two participants even indicated that their confidence level varied based on the type of visualization; both rated their confidence for heatmap answers higher than for trendline answers. The advantages and disadvantages of each visualization became more evident through the interview and the survey. For example, the heatmap was useful in finding global patterns quickly but interpreting values via hue and brightness had split views. In general, participants found the trendline visualization familiar and could observe the extrema more easily. Next we present the participants' preference for trendline and heatmap visualizations.

6.3.1 *Research Q3: Heatmap vs. Trendline*

Participants who preferred the heatmap over the trendline visualization appreciated the extra color dimension of heatmaps. Participants generally found the colorful visualization aesthetically pleasing. P17 commented that the heatmap is "really interesting to watch over time. Especially, at first when things were one or two blocks of color and then to sort of watch the data emerge and to then watch the different boxes become something . . . I actually caught myself watching for a long time." However, the ease of interpreting the colors was debatable; some participants (N=7) stated that colors helped them distinguish the values of the competing groups while others (N=4) found comparing and averaging colors burdensome and not intuitive. It was especially difficult to perceive color differences when the difference of values was small or when the compared blocks were distant on the screen.

Another emerging theme centered about the emergence of easily noticeable color patterns in the early iterations. One participant (P12) commented that the heatmap isolated the interesting patterns faster than the trendline. Although the quick emergence of color patterns is advantageous in making faster decisions, one participant (P8) accurately pointed out the danger of making the decision too soon as it could lead to confirmation bias if the later iterations diverge from the initial color pattern.

The familiarity of the trendline visualization attracted participants with its intuitively interpretable values and easily noticeable changes for consecutive values. Participants also found differentiating high and low points that were distant on a line graph much easier than comparing different the shades of a color in the heatmap visualization. However, the numerous peaks and valleys disturbed some participants as the focal point of the visualization became uncertain. Hovering over a specific point was harder on the trendline to determine exact values since the selection was made solely based on the x-coordinate of the mouse and even a small perturbation to the right would result in a different selection and value.

All but one participant, P16, believed both the heatmap and trendline visualizations were easily interpretable. In the survey, the average usability of INCVISAGE was rated 4.25 out of 5 (σ = .64). Participants also noted easy learning curve for INCVISAGE; all of them felt comfortable with the tool after an hour of use.

6.4 Limitations and Future Work

We identified some limitations and possible future directions. Our approach to approximate visualizations relies heavily on the smoothness of the data; noise and outliers in the dataset impede generating a useful approximation quickly. As highlighted in Section 7.2, when the value of the point of interest and its neighbor(s) is very close, INCVISAGE might choose to reveal that point at later iterations. As a result, any user looking to find quick insights may

Approach	Trendline				Heatmap			
	Extrema Based Questions		Range Based Questions		Extrema Based Questions		Range Based Questions	
	Accuracy	Time (sec)	Accuracy	Time (sec)	Accuracy	Time (sec)	Accuracy	Time (sec)
INCVISAGE	**94.55%**	**25.0638**	**62.50%**	**22.0822**	**83.47%**	**31.6012**	**97.50%**	34.7992
OLA	45.83%	26.4022	52.50%	27.3125	79.13%	**31.3846**	95%	**25.4782**

Table 4: Overall Accuracy and Submission Time Statistics Per Question Category

select an incorrect sub-optimal point. INCVISAGE currently does not offer any explicit guarantee of an answer, which was pointed out as a limitation of the tool by one of the participants (P8). In the next version, we could incorporate a measure of confidence in the tool by noting the variation of values from one iteration to the next. As the groups converge towards the actual value, the segment approximations begin to stabilize, reducing the variation between successive segment approximation.

We discussed the interface issues in Section 6.2. The limitations of the user study fall into three categories—the ambiguity in three quiz questions, interface control, and participant demographics. The ambiguity of the three questions (*R1, R4, and R7*) in the quiz phase led to unintended interface issues (4 issues out of 280 submissions). The result is that participants incorrectly answered questions due to incorrect queries rather than because incorrect interpretation of the visualizations. This highlights the limitations of INCVISAGE with prepared attributes, i.e., if someone downloads a dataset as is and tries to use it with our system, similar ambiguities may occur. From the interface perspective, two participants (P4 and P8) suggested adding axes to the snapshots, which would help them compare values across iterations and in turn, ensure that the approximation is approaching actual values. Participants also desired more control as they explored the data set. One participant (P8) suggested including a command-line interface to allow for more specific queries, while others suggested adding more options, and even different visualization styles. Other interface suggestions that arose included the ability to zoom in and out and to select a specific area from which to sample more. Participants also offered archival suggestions. Two (P10 and P15) participants suggested adding an option to download the final visualization, snapshots, and the data summary. This archival feature would help users explore larger data sets over a longer period of time. Finally, our participant pool demographics do not match the demographics of the general audience intended for this tool. Future studies will reach a larger, more diverse audience.

7. EVALUATION: DECISION MAKING

Previously, we evaluated the interpretability of INCVISAGE, compared trendlines to heatmaps, and qualitatively studied how users felt about the inherent uncertainty. We now compare INCVISAGE with traditional online-aggregation-like [20] approaches (*OLA*) that depict approximations of visualizations as samples are taken (as in the first and third row of Figure 1). Specifically, does INCVISAGE lead to faster and/or more accurate decision making?

7.1 Study Design and Participants

Our study design was similar to our previous study, with four phases, an introduction, a quiz phase, an interview for qualitative data, and finally a closing survey. We introduced the INCVISAGE approach as the "grouped value" approach, and the *OLA* approach as the "single value" approach.

We recruited 20 participants (11 male, 9 female) via a university email newsletter. Our participants included 12 graduate students (4 in computer science), 5 undergraduate students, one researcher, and two university employees. All of the participants had experience with data analysis tools. The median age of the participants was 25 (μ = 26.06; σ = 6.84). Out of the 20 participants, 2 reported that they analyzed or worked with data "Daily", 10 answered "Monthly", 5 "Weekly" while the remaining participants worked rarely. The duration of the sessions ranged from approximately 60 to 75 minutes. Each participant received $10 per hour at the end of their session. All of the studies were conducted in the same lab setting with the same two researchers.

Quiz Phase Design. In designing the quiz phase, we used the flight dataset (details in Section 5), with 20 questions in total—10 on trendlines and 10 on heatmaps. In each case, five questions were reserved for INCVISAGE, and five for *OLA*. We used the same categorizations of questions as in our first study—range-based and extrema-based. These questions are listed in Table 5. As before, we randomized the order of the tools, and the order of the questions.

The Scoring Function. As in Section 6, a score was provided to the user as they answered questions. The score was computed as a product $S = P \cdot A$, where P corresponded to how quickly the user made a decision, and A to the accuracy. The formulae for A were similar to the previous study. Instead of setting P to be proportional to the number of remaining iterations, here, in order to allow the scores to be comparable between *OLA* and INCVISAGE, we set P to be $\frac{T-t}{T}$, where T is the time taken to compute the visualization by scanning the entire dataset, while t is the time taken by the user.

7.2 Quantitative Analysis of the Quiz Phase

In discussing the participants' quiz performance, we investigate both their accuracy (using A above) as well as answer submission time for both INCVISAGE and *OLA*.

Summary: For trendlines, INCVISAGE exhibits a 62.52% higher accuracy than *OLA* for both question types, while also reducing the submission time by 10.83%. For heatmaps, INCVISAGE exhibits 4.05% higher accuracy than *OLA* for both question types. The submission time for range-based questions with INCVISAGE is higher than *OLA*.

Accuracy and Submission Time Statistics. Table 4 summarizes the overall accuracy and the submission times for both INCVISAGE and *OLA*. For trendlines, INCVISAGE outperformed *OLA* in terms of both accuracy and submission times. For extrema based questions, the overall accuracy of INCVISAGE was almost double than that of *OLA*. The submission times were also lower for INCVISAGE for both types of questions. Overall, users are able to make *faster* and *more accurate* decisions using INCVISAGE than *OLA*. There is a dip in the overall accuracy for the range based questions for both approaches. Since the accuracy of the range based questions was either 0 or 1, any incorrect submission incurred a higher penalty, thereby reducing overall accuracy.

For heatmaps, INCVISAGE exhibited better accuracy than *OLA*—in particular, an improvement of 4.05%. For extrema based questions, the submission times were almost equal. However, for range based questions submissions with INCVISAGE took longer than *OLA*. We found that when using INCVISAGE with heatmaps, participants waited for the larger blocks to break up in order to compute averages over ranges, thereby increasing submission times but providing more accurate answers. As it turns out, the initial (larger) blocks in INCVISAGE do provide average estimates across ranges and could have been used to provide answers to the questions quickly. The unfamiliarity with incremental heatmap visualizations, and heatmaps in general, contributed to this delay. In hindsight, we could have educated the participants more about how to draw conclusions from the INCVISAGE heatmaps and this may have reduced submission times.

Per Question Statistics. Figure 15 shows the dot graph analysis of accuracy for the extrema based questions for both visualization types. The submissions using INCVISAGE and *OLA* are highlighted by "cyan" and "magenta" circles, respectively. The x-axis represents the accuracy while the y-axis represents the questions. For trendlines (Figure 15a), majority (99%) of the submissions with INCVISAGE were in close proximity of the correct answer,

Questions	Trendlines	Heatmaps
1	Which day of the year enjoys the shortest Arrival Delay?	Which Week of Year had the highest Departure Delay?
2	Find the day with the highest carrier delay for US airways.	Among the following three months, which of month has the highest number of Days (of Month) with a high Arrival Delay?A) FEB B) JUL C) OCT
3	Overall, the Arrival Delay is the worst/highest in- A) JAN 01-JAN 15 B) JUL 11- JUL 25 C) DEC 16 - DEC 31	Which of the following Months exhibits a higher Arrival Delay for the majority of the Years?A) JAN B) JUL C) NOV
4	Which day of the year shows a higher Departure Delay on average for the flights departing the LGA airport?	Find the (Day of Month, Month) pair that had the highest Aircraft Delay.
5	Which of the following ranges contains the day with the highest Aircraft Delay for the UA aircrafts- A) JAN 01- JAN 10 B) JUN 10- JUN 20 C) DEC 21- DEC 31	Which Year contains the Day of Week with the highest Departure Delay?
6	Which Day of the year has the highest Departure Delay?	Find the (Day of Month, Month) pair in the year 2013, that had the highest Weather Delay.
7	In the city of Atlanta-GA, which Day of the year has the highest NAS (National Aviation System) Delay?	Which of the following Months exhibited a lower Departure Delay for the majority of the Years?A) FEB B) JUN C) SEP
8	In the city of Chicago-IL, bad weather forces longer weather delays on average in the following period- A) Day 1-15 B) Day 190-205 C) Day 350-366	Which of (Day of Month, Month) pair has the highest Arrival Delay?
9	Which day of the year 2010 enjoys the shortest Arrival Delay?	Which Week of Year contains the Day of Week with the highest Arrival Delay?
10	In the Year 2011, which of the following months have lower Security Delays? A) JAN B) JUL C) NOV	Which of the following Years contain the month with the highest Security Delay? A) 2004 B) 2008 C) 2012 D) 2015

Table 5: User study questions.

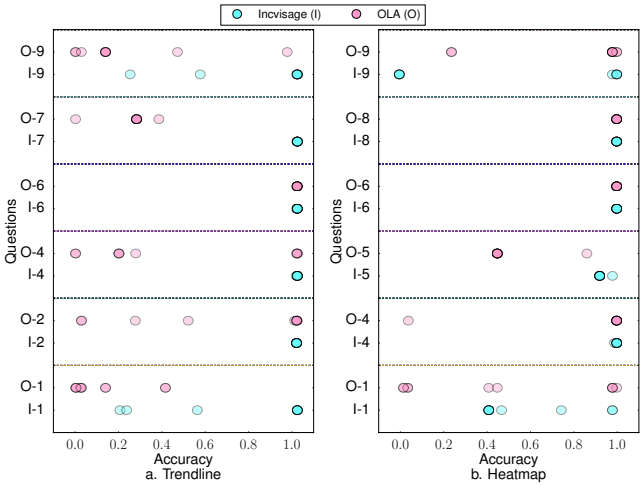


Figure 15: Accuracy statistics of extrema based questions for (a) trendline and (b) heatmap visualizations.

whereas with *OLA* the accuracy drops quite a lot—only 55% of the submissions were in close proximity with the correct answer. For heatmaps (Figure 15b), again there are more submissions with INCVISAGE (80%) that are in close proximity of the correct answer compared to *OLA* (52%). Figure 16 shows the accuracy for the range based questions for both visualization types. The y-axis represents the accuracy while the x-axis represents the questions. The submissions using INCVISAGE and *OLA* are highlighted by "cyan" and "magenta" bars, respectively. For trendlines (Figure 16a), none of the submissions for Q5 was correct. For the rest of the questions, submissions with INCVISAGE had higher accuracy than *OLA*. For heatmaps (Figure 16b), accuracy of INCVISAGE is only slightly better than *OLA*.

Submission Trends. Figure 17 plots the accuracy of all the submissions according to the scoring functions described in Section 7.1. The x-axis represents submission time in seconds; *accuracy* appears on the y-axis. For both trendline (Figure 17a) and heatmap (Figure 17b) visualizations, participants opted to submit their answers as quickly as possible for both the approaches, i.e., the participants chose to trade accuracy for time.

7.3 Interview and Survey Phase

In this section, we present a qualitative analysis of the participants' perceptions of both the approaches based on their interview

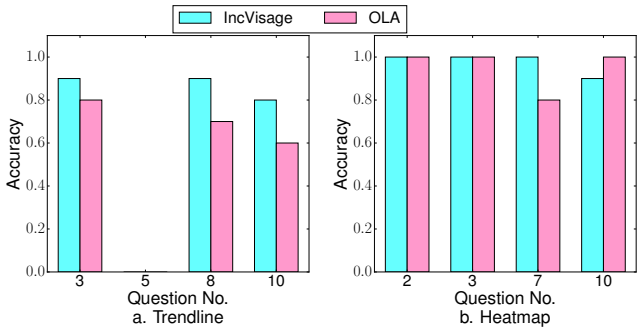


Figure 16: Accuracy statistics of extrema based questions for

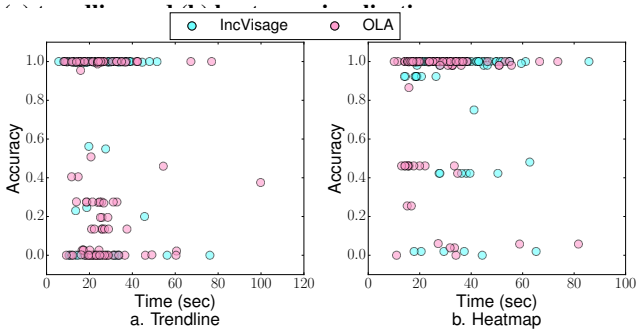


Figure 17: Accuracy vs Submission Time statistics for (a) trendline and (b) heatmap visualizations.

and survey responses.

Summary: Participants were reasonably confident when they submitted answers for both visualization types using INCVISAGE. The trendline visualization using *OLA* was unstable and had low interpretability that resulted in participants submitting answers with low confidence. Majority of the participants preferred the INCVISAGE representations for both visualizations.

Interview. We conducted semi-structured interviews to gauge our participants' motivations for stopping the visualization at a certain iteration, and their confidence on their answers (for both INCVISAGE and *OLA*). The main motivations for terminating a visualization were the emergence of obvious extrema ($N = 10$), gaining sufficient confidence in an answer ($N = 6$), or both ($N = 4$). When asked to rate their confidence at the time of submission on a scale of 1 to 10 (10 being the most confident), we had varied responses depending on the visualization types. For trendlines, participants were reasonably confident ($\mu = 6.53$, $\sigma = 1.89$) when

using INCVISAGE, but much less confident ($\mu = 4.44$, $\sigma = 1.27$) when using *OLA*. For heatmaps, participants were slightly more confident when using *OLA* ($\mu = 7.15$, $\sigma = 0.86$) than when using INCVISAGE ($\mu = 6.76$, $\sigma = 1.97$). Majority of the participants ($N = 3$) who preferred *OLA* liked the fact that they were able to see each individual data point at all times. INCVISAGE on the other hand, hid information early on by approximating the visualizations which was less desirable to them. Among 20 participants, majority ($N = 12$) preferred the INCVISAGE representation over the *OLA* ($N = 6$) representation of the visualizations while two participants equally preferred the two approaches.

When using INCVISAGE. participants were able to interpret the initial high level approximations to identify specific regions of interest and eventually found the answer. On the other hand, they thought *OLA* to be unstable and difficult to interpret. One of the participants (P14) said the following—"For INCVISAGE, it was easier to know when I wanted to stop because I had the overall idea first. And then I was just waiting to get the precise answer because I knew it was coming. So it was the difference. *OLA*, it was a shot in the dark where I see a little bit where it is, I would wait to see if it stays as the answer". Another participant (P15) also expressed similar observations—"With single values, there was just so much going on I was like 'OK, where am I trying to focus on. What is either the highest or the lowest?' versus the grouped values, it started out really simple and then became more complex to be able to show the differences". The same participant also preferred the aesthetics of INCVISAGE—"I preferred the grouped (one), because it made it easier to kind of narrow down at least the range. So if you didn't know the exact date, you could at least be close. Versus with the single value, it could, there could be two values that look similar and if you picked the wrong one, you were very wrong, potentially."

Survey. The survey consisted of four Likert scale questions for each visualization type to measure the interpretability and the usability of the competing approaches: INCVISAGE and *OLA*. Also, for each visualization type, there were two free-form questions asking the participants to list the positive and negative aspects of both INCVISAGE and *OLA*. For trendlines, INCVISAGE received higher average ratings (out of 5) for interpretability (INCVISAGE: μ = 3.93; σ = 0.92; *OLA*: μ = 2.67; σ = 1.29) and satisfaction levels (INCVISAGE: μ = 3.93; σ = 0.88; *OLA*: μ = 2.67; σ = 0.82). On the other hand, for heatmaps, *OLA* received slightly better average ratings for interpretability (INCVISAGE: μ = 3.73; σ = 0.80; *OLA*: μ = 4.00; σ = 0.76) and satisfaction levels (INCVISAGE: μ = 3.60; σ = 0.99; *OLA*: μ = 3.87; σ = 0.64).

Limitations and Future Work. Similar to our first study, users again mentioned uncertainty as an issue when submitting an answer. For the heatmap representation of INCVISAGE the confidence of the users could have further boosted if we had a measure of confidence for the visualizations represented which in turn could have further improved the submission times. This could be an interesting direction for future work. One possible approach is to use both INCVISAGE and *OLA* representations side by side so that users can gain further confidence by seeing individual blocks alongside the larger blocks.

8. RELATED WORK

In this section, we review papers from multiple research areas and explain how they relate to INCVISAGE.

Approximate Query Processing (AQP). AQP schemes can operate online, i.e., select samples on the fly, and offline, i.e, select samples prior to queries being issued. Among online schemes, certain approaches respect a predefined accuracy constraint for computing certain fixed aggregates without indices [22, 23], and with indexes [19, 36]. The objectives and techniques are quite different from that of incrementally improving visualizations. Offline AQP systems [9, 8, 11, 18] operate on precomputed samples. Garofalakis et al. [18] provides a detailed survey of the work in this space. Unlike these approaches, INCVISAGE deals with ad-hoc visualizations.

Approximate Visualization Algorithms. We have already discussed IFOCUS [32], PFunkH [10] and ExploreSample [53] in the introduction section. IFOCUS [32], PFunk-H [10], and ExploreSample [53] are online approximate visualization algorithms. IFOCUS [32] generates bar charts preserving ordering guarantees between bars quickly, but approximately. PFunk-H [10] uses perceptual functions to provide approximate answers that differ from the true answers by a perceptually indiscernible amount. ExploreSample [53] is also an online sampling algorithm that deals with 2d scatterplots coupled with heatmaps; approximating the visualization while preserving the outliers and the overall distribution. Unlike these one-shot approaches, we generate visualizations that incrementally improve over time. Concurrent work has shown that analysts are willing to use approximate visualizations in real data exploration scenarios [39]. This work introduces an optimistic visualization system that allows users to explore approximate visualizations and verify the results of any visualization they feel uncertain about at a later time. The visualization to be verified is computed in the background while user continues to explore the data. INCVISAGE's approach can be combined with this approach, since verification of decisions made using approximate visualizations may be valuable (and thus the two directions provide orthogonal and complementary benefits).

Incremental Visualization. We have already discussed Online aggregation [20] and sampleAction [17] in the introduction section. Online aggregation [20] introduced the idea of incremental data analysis, while sampleAction [17] uses online aggregation for displaying incremental visualizations. Jermaine et al. [27, 28] further explored incremental database queries. Online aggregation places the onus on the users to regulate the sampling of the groups—instead INCVISAGE automates the sampling process, and produces smooth refinements across iterations. Recent user studies by Zgraggen et al. [54] demonstrate that an *OLA*-style system outperforms one-shot computation of the visualization in terms of number of insights gained—they do not contribute any new algorithms, however. In our work, we additionally demonstrate that INCVISAGE reduces the number of user mistakes made in decision making compared to *OLA*.

Bertolotto et al. [13] uses a multiresolution spatial maps construction technique [41] to compute representations of spatial maps at various resolutions offline to reduce time taken for storage and therefore data transfer at each resolution, while preserving geographical features of interest including the size and interaction of points, lines and shapes—this is therefore a offline data compression approach for a fixed spatial map. INCVISAGE, on the other hand, uses sampling to compute the k-increments of visualizations online and can support ad-hoc queries.

Visualization Tools. In recent years, several interactive visualization tools have been introduced [44, 48, 31, 46, 42, 43]. The algorithms provided in this paper can be incorporated in these tools so that users can quickly identify key features of data.

Scalable Visualizations. A number of recent tools support scalable visualization generation [37, 30, 35] by precomputing and storing aggregates—this can be prohibitive on datasets with many attributes. M4 [29] uses rasterization parameters to reduce the dimensionality of a trendline at query level—selects the groups that correctly represent the distributions. INCVISAGE on the other hand, reveals features of visualizations in the order of prominence for arbitrary queries.

Approximation of Distributions. Approximating a data distribu-

tion by histograms has also been studied previously [7, 24, 26]. These methods do not sample iteratively from groups—they operate in a one-shot manner, and focus only on COUNT queries. Donjerkovic et al. [16] maintains histograms over evolving data, once again for COUNT queries.

9. CONCLUSIONS

We introduced the notion of incrementally improving visualizations and demonstrated that our incremental visualization tool, INCVISAGE, helps users gain insights and make decisions quickly. On very large datasets, INCVISAGE is able to achieve a $46\times$ speedup relative to *SCAN* in revealing the first 10 salient features of a visualization with suitable error guarantees that are comparable to a dynamic programming approach, but without a high computational overhead. Our user studies demonstrate that users chose to trade accuracy for time to make rapid decisions, that too at higher accuracy than traditional approaches. There are a number of interesting future directions, such as modeling and displaying the degree of uncertainty, along with a wider range of operations (e.g. pausing at segment level or group level), and alternative views (e.g., overlaying incremental visualizations and traditional approaches). Finally, gaining a better understanding of the sorts of decisions for which one-shot approaches and incremental visualization approaches are appropriate is a promising future direction.

10. REFERENCES

[1] Intel sensor dataset. http://db.csail.mit.edu/labdata/labdata.html.

[2] Microsoft's power bi hits 5m subscribers, adds deeper excel integration. http://www.pcworld.com/article/3047083/. Accessed: 05-22-2016.

[3] Nyc taxi dataset. http://publish.illinois.edu/dbwork/open-data/.

[4] Tableau q2 earnings: Impressive growth in customer base and revenues. http://www.forbes.com/sites/greatspeculations/2015/07/31/tableau-q2-earnings-impressive-growth-in-customer-base-and-revenues.

[5] Us flight dataset. http://stat-computing.org/dataexpo/2009/the-data.html.

[6] Wunderground weather dataset. https://www.wunderground.com/.

[7] J. Acharya et al. Fast and near-optimal algorithms for approximating distributions by histograms. In *PODS*, pages 249–263. ACM, 2015.

[8] S. Acharya et al. The aqua approximate query answering system. In *SIGMOD Rec.*, volume 28, pages 574–576. ACM, 1999.

[9] S. Agarwal et al. Blinkdb: queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42. ACM, 2013.

[10] D. Alabi et al. Pfunk-h: Approximate query processing using perceptual models. In *HILDA Workshop*, pages 10:1–10:6. ACM, 2016.

[11] B. Babcock et al. Dynamic sample selection for approximate query processing. In *SIGMOD Conf.*, pages 539–550. ACM, 2003.

[12] Z. Bar-Yossef. *The Complexity of Massive Data Set Computations*. PhD thesis, Berkeley, CA, USA, 2002. AAI3183783.

[13] M. Bertolotto et al. Progressive vector transmission. In *Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 152–157. ACM, 1999.

[14] L. Cam and G. Yang. *Asymptotics in Statistics: Some Basic Concepts*. Springer Series in Statistics. Springer New York, 2000.

[15] M. Correll et al. Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE TVCG*, 20(12):2142–2151, 2014.

[16] D. Donjerkovic et al. Dynamic histograms: Capturing evolving data sets. In *ICDE'00*, pages 86–86. IEEE Computer Society Press; 1998, 2000.

[17] D. Fisher et al. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *CHI'12*, pages 1673–1682. ACM, 2012.

[18] M. N. Garofalakis. Approximate query processing: Taming the terabytes. In *VLDB*, 2001.

[19] P. J. Haas et al. Selectivity and cost estimation for joins based on random sampling. *Journal of Computer and System Sciences*, 52(3):550–569, 1996.

[20] J. M. Hellerstein et al. Online aggregation. *SIGMOD Rec.*, 26(2), 1997.

[21] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[22] W.-C. Hou et al. Statistical estimators for relational algebra expressions. In *PODS*, pages 276–287. ACM, 1988.

[23] W.-C. Hou et al. Processing aggregate relational queries with hard time constraints. In *SIGMOD Rec.*, volume 18, pages 68–77. ACM, 1989.

[24] P. Indyk et al. Approximating and testing k-histogram distributions in sub-linear time. In *PODS*, pages 15–22. ACM, 2012.

[25] F. Jackson. The basic gamma-function and the elliptic functions. *Proceedings of the Royal Society of London.*, 76(508):127–144, 1905.

[26] H. V. Jagadish et al. Optimal histograms with quality guarantees. In *VLDB*, volume 98, pages 24–27, 1998.

[27] C. Jermaine et al. The sort-merge-shrink join. *ACM Transactions on Database Systems (TODS)*, 31(4):1382–1416, 2006.

[28] S. Joshi et al. Materialized sample views for database approximation. *IEEE TKDE*, 20(3):337–351, 2008.

[29] U. Jugel et al. M4: a visualization-oriented time series data aggregation. *VLDB Endow.*, 7(10):797–808, 2014.

[30] S. Kandel et al. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *AVI*, pages 547–554. ACM, 2012.

[31] A. Key et al. Vizdeck: self-organizing dashboards for visual analytics. In *SIGMOD Conf.*, pages 681–684. ACM, 2012.

[32] A. Kim et al. Rapid sampling for visualizations with ordering guarantees. *VLDB*, 8(5):521–532, 2015.

[33] A. Kim et al. Speedy browsing and sampling with needletail. Technical report, 2016. https://arxiv.org/abs/1611.04705.

[34] N. Koudas. Space efficient bitmap indexing. In *CIKM*, pages 194–201, 2000.

[35] L. Lins et al. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE TVCG*, 19(12):2456–2465, 2013.

[36] R. J. Lipton et al. Efficient sampling strategies for relational database operations. *Theoretical Computer Science*, 116(1):195–226, 1993.

[37] Z. Liu et al. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[38] Z. Liu et al. The effects of interactive latency on exploratory visual analysis. *IEEE TVCG*, 20(12):2122–2131, 2014.

[39] D. Moritz et al. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *CHI*, 2017.

[40] S. G. Perlman. System and method for rendering graphics and video on a display, June 26 2007. US Patent 7,236,204.

[41] E. Puppo et al. Towards a formal model for multiresolution spatial maps. In *Advances in Spatial Databases*, pages 152–169. Springer, 1995.

[42] T. Siddiqui et al. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *VLDB Endowment*, 10(4):457–468, 2016.

[43] T. Siddiqui et al. Fast-forwarding to desired visualizations with zenvisage. CIDR, 2017.

[44] C. Stolte et al. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE TVCG*, 8(1):52–65, 2002.

[45] K. Stromberg. *Probability for analysts*. CRC Press, 1994.

[46] M. Vartak et al. Seedb: efficient data-driven visualization recommendations to support visual analytics. *VLDB Endow.*, 8(13):2182–2193, 2015.

[47] Weisstein, Eric W, Wolfram Research, Inc. Euler-mascheroni constant. 2002.

[48] R. Wesley et al. An analytic data engine for visualization in tableau. In *SIGMOD Conf.*, pages 1185–1194. ACM, 2011.

[49] M. B. Wilk et al. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17, 1968.

[50] A. P. Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP*, volume 9, pages 150–153. IEEE, 1984.

[51] K. Wu et al. Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems (TODS)*, 31(1):1–38, 2006.

[52] K. Wu et al. Analyses of multi-level and multi-component compressed bitmap indexes. *ACM Transactions on Database Systems (TODS)*, 35(1):2, 2010.

[53] Y. Wu et al. Efficient evaluation of object-centric exploration queries for visualization. *VLDB*, 8(12):1752–1763, 2015.

[54] E. Zgraggen et al. How progressive visualizations affect exploratory analysis. *IEEE TVCG*, 2016.

APPENDIX

A. PROBLEM FORMULATION (HEATMAPS)

In this section, we describe the concepts in the context of our visualizations in row 4 of Figure 1, i.e, incrementally improving heatmap visualizations.

Blocks and b_k-Block Approximations. We denote the cardinality of our group-by dimension attributes X_a as m and X_b as n, i.e., $|X_a| = m$ and $|X_b| = n$. In Figure 1, $X_a = 10$ and $X_b = 11$. Over the course of visualization generation, we display one value of AVG(Y) corresponding to each combination of groups $x_i \in X_a, i \in 1 \ldots m$ and $x'_j \in X_b, j \in 1 \ldots n$—thus, the user is always shown a complete heatmap visualization. We call the pair (x_i, x'_j) as *group combination* and denote by $x_{i,j}$. To approximate our heatmaps, we use the notion of *blocks* that encompass one or more *group combinations*. We define a block as follows:

Definition 4. *A* block β *corresponds to a pair* $(I \times J, \eta)$*, where* η *is a value, while* I *is an interval* $I \subseteq [1, m]$ *spanning a consecutive sequence of groups* $x_i \in X_a$ *and* J *is an interval* $J \subseteq [1, n]$ *spanning a consecutive sequence of groups* $x'_j \in X_b$*. The block encompasses the* $m \times n$ *group combinations* $x_{i,j}$ *where* $x_i \in I$ *and* $x_j \in J$*.*

For example, the block β $([2, 4] \times [1, 2], 0.7)$ has a value of 0.7 and encompasses the group combinations $x_{2,1}$, $x_{2,2}$, $x_{3,1}$, $x_{3,2}$, $x_{4,1}$, $x_{4,2}$. Then, a b_k-block approximation of a visualization comprises b_k disjoint blocks that span the entire range of x_i, $i = 1 \ldots m$ and x_j, $j = 1 \ldots n$. We explain the value of b_k later. Formally, a b_k-block approximation is defined as follows:

Definition 5. *A* b_k-block *approximation is a tuple* $M_k = (\beta_1, \ldots, \beta_{b_k})$ *such that the block* $\beta_1, \ldots, \beta_k$ *partition the interval* $[1, m] \times [1, n]$ *into* b_k *disjoint sub-intervals.*

As an example from Figure 1, at t_2, we are displaying a 4-block approximation, comprising four blocks $([1, 6] \times [1, 9], 0.8)$, $([7, 10] \times [1, 9],\ 0.4)$,$([1, 6] \times [10, 11], 1.0)$, and $([7, 10] \times [10, 11],\ 1.4)$. When the number of blocks is unspecified, we simply refer to this as a *block* approximation.

Incrementally Improving Visualizations. An *incrementally improving visualization* is defined to be a sequence of block approximations, $m \times n$ in total, $(M_1, \ldots, M_{m \times n})$, where the ith item M_i in the sequence is a b_i-block approximation, and is formed by selecting one of the block in the b_{i-1}-block approximation M_{i-1} (the preceding one in the sequence), and dividing that block into either two or four blocks.

Similar to trendlines each block approximation is a *refinement* of the previous, revealing new *features* of the visualization and is formed by dividing one of the block β in the b_i-block approximation into either two or four new blocks to give an b_{i+1}-block approximation: we call this process *splitting a block*. The group combination within $\beta \in M_i$ immediately following which the split occurs is referred to as a *split group combination*. Any group combination in the interval $I \times J \in \beta$ except for the last one can be chosen as a *split group combination*. Since an existing block can be split into either two or four blocks, then $k \leq b_k \leq (3k - 2)$ where $(b_k - k)\%2 = 0$. As an example, in Figure 1, the entire fourth row corresponds to an incrementally improving visualization, where, for example, the 4-block approximation ($k = 2$ and $b_2 = 4$ where, $2 \leq b_2 \leq 4$ and $(b_2 - 2)\%2 = 0$) is generated by taking the block in the 1-block approximation corresponding to $([1, 10] \times [1, 11], 0.5)$, and splitting it at group combination $x_{6,9}$ to give $([1, 6] \times [1, 9], 0.8)$, $([7, 10] \times [1, 9],\ 0.4)$,$([1, 6] \times [10, 11], 1.0)$, and $([7, 10] \times [10, 11], 1.4)$. Therefore, the *split group combination* is $x_{6,9}$ consisting of the pair (6,9). The reasoning behind imposing such restriction was discussed in Section 2.2.

Underlying Data Model and Output Model. We represent the heatmap visualization as a sequence of $m \times n$ distributions $D_{1,1}, \ldots, D_{m,n}$ with means $\mu_{1,1}, \ldots, \mu_{m,n}$ where, $\mu_{i,j} = \text{AVG(Y)}$ for $x_i \in X_a$ and $x_j \in X_b$. To generate our incrementally improving visualization and its constituent block approximations, we draw samples from distributions $D_{1,1}, \ldots, D_{m,n}$. When drawing samples from the distribution $D_{i,j}$, our sampling engine retrieves a sample only when it satisfies the condition $X_a = x_i \wedge X_b = x'_j$ (see Section 2.1). Our goal is to approximate the mean values $\mu_{1,1}, \ldots, \mu_{m,n}$ where, $\mu_{i,j}$ while taking as few samples as possible.

The output of a b_k-block approximation M_k can be represented alternately as a sequence of values $(\nu_{1,1}, \ldots, \nu_{m,n})$ such that $\nu_{i,j}$ is equal to the value corresponding to the block that encompasses $x_{i,j}$. By comparing $(\nu_{1,1}, \ldots, \nu_{m,n})$ with $\mu_{1,1}, \ldots, \mu_{m,n}$, we can evaluate the error corresponding to a b_k-block approximation, as we describe next.

Error. We define the ℓ_2 *squared error* of a b_k-block approximation M_k with output sequence $(\nu_{1,1}, \ldots, \nu_{m,n})$ for the distributions $D_{1,1}, \ldots, D_{m,n}$ with means $\mu_{1,1}, \ldots, \mu_{m,n}$ as

$$\text{err}(M_k) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (\mu_{i,j} - \nu_{i,j})^2 \tag{4}$$

A.1 Problem Statement

Now we formally define the problem for heatmaps which is similar to Problem 1:

Problem 2 (Heatmap)**.** *Given a query* Q_H*, and the parameters* δ, ϵ*, design an incrementally improving visualization generation algorithm that, at each iteration* k*, returns a* b_k*-block approximation while taking as few samples as possible, such that with probability* $1-\delta$*, the error of the* b_k*-block approximation* M_k *returned at iteration* k *does not exceed the error of the best* b_k*-block approximation formed by splitting a block of* M_{k-1} *by no more than* ϵ*.*

B. ALGORITHM FOR HEATMAPS

In this section, we build up our solution to the incrementally improving visualization generation algorithm for heatmaps, *ISPlit-Grid*. We present the major ideas, concepts and proofs required to explain *ISPlit-Grid*.

B.1 Case 1: The Ideal Scenario

We first consider the ideal scenario when the means of the distributions are known (see Section 3.1. In the context of heatmaps, when the means of the distributions are known, the task reduces to identifying the block β_i, splitting which will minimize the b_{k+1}-block approximation M_{k+1}. We now dive straight into the definition of the ℓ_2-squared error of a block. The ℓ_2 *squared error* of a block β_i $(I_i \times J_i, \eta_i)$, where $I_i = [p, q]$, $J_i = [r, s]$ $(1 \leq p \leq q \leq m$ and $1 \leq r \leq s \leq n)$, approximating the distributions $D_{p,r}, \ldots, D_{q,s}$ with means $\mu_{p,r}, \ldots, \mu_{q,s}$ is

$$\begin{aligned}\text{err}(\beta_i) &= \frac{1}{(q - p + 1) \times (q - p + 1)} \sum_{j=p}^{q} \sum_{j'=r}^{s} (\mu_{j,j'} - \eta_i)^2 \\ &= \frac{1}{|\beta_i^I| \times |\beta_i^J|} \sum_{j \in \beta_i^I} \sum_{j' \in \beta_i^J} (\mu_{j,j'} - \eta_i)^2\end{aligned}$$

For the ideal scenario, we can rewrite the expression for $\text{err}(\beta_i)$ as follows:

$$\text{err}(\beta_i) = \frac{1}{|\beta_i^I| \times |\beta_i^J|} \sum_{j \in \beta_i^I} \sum_{j' \in \beta_i^J} \mu_{j,j'}^2 - \mu_{\beta_i}^2 \tag{5}$$

Then, using Equation 4, we can express the ℓ_2 *squared error* of the b_k-block approximation M_k as follows:

$$\text{err}(M_k) = \sum_{i=1}^{b_k} \frac{|\beta_i^I| \times |\beta_i^J|}{mn} \text{err}(\beta_i)$$

Now, let's assume M_{k+1} is obtained by splitting a block $\beta_i \in M_k$ into four blocks T, U, V, and W. Then, the error of M_{k+1} is:

$$\begin{aligned}\text{err}(M_{k+1}) &= \text{err}(M_k) - \frac{|\beta_i^I| \times |\beta_i^J|}{mn}\text{err}(\beta_i) \\ &+ \frac{|T^I||T^J|}{m}\text{err}(T) + \frac{|U^I||U^J|}{mn}\text{err}(U) \\ &+ \frac{|W^I||W^J|}{mn}\text{err}(V) + \frac{|V^I||V^J|}{mn}\text{err}(W) \\ &= .\text{err}(M_k) + \frac{|\beta_i^I| \times |\beta_i^J|}{mn}\mu_{\beta_i}^2 \\ &- \frac{|T^I||T^J|}{m}\mu_T^2 - \frac{|U^I||U^J|}{mn}\mu_U^2 \\ &- \frac{|V^I||V^J|}{mn}\mu_V^2 - \frac{|W^I||W^J|}{mn}\mu_W^2\end{aligned}$$

We use the above expression to define the notion of *improvement potential* for heatmaps. The *improvement potential* of a block $\beta_i \in M_k$ is the minimization of the error of M_{k+1} obtained by splitting β_i into T,U,V and W. Thus, the *improvement potential* of block β_i relative to T,U,V and W is

$$\begin{aligned}\Delta(\beta_i, T, U, V, W) = &\frac{|T^I||T^J|}{mn}\mu_T^2 + \frac{|U^I||U^J|}{mn}\mu_U^2 \\ &\frac{|V^I||V^J|}{mn}\mu_V^2 + \frac{|W^I||W^J|}{mn}\mu_W^2 - \\ &\frac{|\beta_i^I| \times |\beta_i^J|}{mn}\mu_{\beta_i}^2\end{aligned}$$

Now, The *split group combination* maximizing the *improvement potential* of β_i, minimizes the error of M_{k+1}. Therefore, the maximum *improvement potential* of a block is expressed as follows:

$$\Delta^\star(\beta_i) = \max_{T,U,V,W \subseteq \beta_i} \Delta(\beta_i, T, U, V, W)$$

Lastly, we denote the *improvement potential* of a given M_{k+1} by $\phi(M_{k+1}, \beta_i, T, U, V, W)$, where $\phi(M_{k+1}, \beta_i, T, U, V, W)$ = $\Delta(\beta_i, T, U, V, W)$. Therefore, the maximum *improvement potential* of M_{k+1}, $\phi^\star(M_{k+1})$ =max$_{\beta_i \subseteq M_k}$ $\Delta^\star(\beta_i)$. When the means of the distributions are known, at iteration $(k+1)$, the optimal algorithm simply selects the refinement corresponding to $\phi^\star(L_{k+1})$, which is the block approximation with the maximum improvement potential.

2.2 Case 2: The Online-Sampling Scenario

In the online sampling scenario, the means of the distributions are unknown. Similar to trendlines, we describe our approach for selecting a refinement at a single iteration assuming samples have already been drawn from the distributions. Then, we describe our approach for selecting samples.

2.2.1 Selecting the Refinement Given Samples

In the online sampling scenario, we define a new notion of error err' and optimize for that error (see Section 3.2). Since, we draw samples from the distributions, we can obtain the *estimated improvement potential* as follows:

$$\begin{aligned}\widetilde{\phi}(M_{k+1}, \beta_i, T, U, V, W) = &\frac{|T^I||T^J|}{mn}\widetilde{\mu}_T^2 + \frac{|U^I||U^J|}{mn}\widetilde{\mu}_U^2 \\ &\frac{|V^I||V^J|}{mn}\widetilde{\mu}_V^2 + \frac{|W^I||W^J|}{mn}\mu_W^2 - \\ &\frac{|\beta_i^I| \times |\beta_i^J|}{mn}\mu_{\beta_i}^2\end{aligned}$$

At iteration $(k+1)$, all the blocks in M_k and all the segments that may appear in M_{k+1} after splitting a block are called *boundary blocks*. In the following theorem, we show that if we estimate the boundary blocks accurately, then we can find a split which is very close to the best possible split. We can prove the following for heatmaps:

Theorem 8. *If for every boundary block T of the b_k-block approximation M_k, we obtain an estimate $\tilde{\mu}_T$ of the mean μ_T that satisfies*

$$\left|\tilde{\mu}_T^2 - \mu_T^2\right| \leq \frac{\epsilon mn}{10|T^I||T^J|},$$

then the refinement $M_{k+1}^\dagger$ of M_k that minimizes the estimated error $\widetilde{\text{err}}(M_{k+1}^\dagger)$ will have error that exceeds the error of the best refinement M_{k+1}^* of M_k by at most

$$\text{err}(M_{k+1}^\dagger) - \text{err}(M_{k+1}^*) \leq \epsilon.$$

Proof. The *estimated improvement potential* of the refinement L_{k+1} satisfies

$$|\widetilde{\phi}(M_{k+1}) - \phi(M_{k+1})| \leq \frac{\epsilon}{2}.$$

We can obtain this inequality by using the expression for $\widetilde{\phi}(M_{k+1})$ and $\phi(M_{k+1})$, and substituting the terms like $\left|\tilde{\mu}_T^2 - \mu_T^2\right|$ with $\frac{\epsilon mn}{10|T^I||T^J|}$. Together this inequality, the identity $\text{err}(M_{k+1}) = \text{err}(M_k) - \phi(M_{k+1})$, and the inequality $\phi(M_{k+1}) \leq \phi(M_{k+1}^\dagger)$ imply that

$$\begin{aligned}&\text{err}'(M_{k+1}^\dagger) - \text{err}'(M_{k+1}^*) \\ &= \phi(M_{k+1}^*) - \phi(M_{k+1}^\dagger) \\ &= \phi(M_{k+1}^*) - \widetilde{\phi}(M_{k+1}^*) + \widetilde{\phi}(M_{k+1}^*) - \phi(M_{k+1}^\dagger) \\ &\leq \phi(M_{k+1}^*) - \widetilde{\phi}(M_{k+1}^*) + \widetilde{\phi}(M_{k+1}^\dagger) - \phi(M_{k+1}^\dagger) \\ &\leq \epsilon.\end{aligned}$$

□

2.2.2 Determining the Sample Complexity

To achieve the error guarantee for Theorem 8, we need to retrieve a certain number of samples from each of the distributions $D_{1,1}, \ldots, D_{m,n}$. Similar to trendlines, we again assume that the data is generated from a sub-gaussian distribution. Given the generative assumption, we can determine the number of samples required to obtain an estimate with a desired accuracy using Hoeffding's inequality [21] which leads us to the following Lemma:

Lemma 5. *For a fixed $\delta > 0$ and a b_k-block approximation of the distributions $D_{1,1}, \ldots, D_{m,n}$ represented by $m \times n$ independent random samples $x_{1,1}, \ldots, x_{m,n}$ with sub-Gaussian parameter σ^2 and mean $\mu_{i,j} \in [0, a]$ if we draw $C = \left\lceil \frac{800\, a\, \sigma^2}{\epsilon^2 mn} \ln\left(\frac{4mn}{\delta}\right) \right\rceil$ samples uniformly from each $x_{i,j}$, then with probability at least $1-\delta$, $\left|\widetilde{\mu}_T^2 - \mu_T^2\right| \leq \frac{\epsilon mn}{10|T^I||T^J|}$ for every boundary block T of M_k.*

Proof. Fix any boundary block T contained in the block $\beta_i \in L_k$. Then, we draw samples $x_{i,j,1}, x_{i,j,,2}, \ldots, x_{i,j,,C}$ uniformly from $x_{i,j}$ such that $x_i \in T^I$ and $x_j \in T^J$, then

$$\begin{aligned}\widetilde{\mu}_T - \mu_T &= \frac{1}{C|T^I||T^J|}\sum_{i \in T^I}\sum_{j \in T^J}\sum_{g=1}^{C} x_{i,j,g} - \frac{1}{|T^I||T^J|}\sum_{i \in T^I}\sum_{j \in T^J}\mu_{i,j} \\ &= \frac{1}{C|T^I||T^J|}\sum_{i \in T^I}\sum_{j \in T^J}\sum_{g=1}^{C}(x_{i,j,g} - \mu_{i,j}).\end{aligned}$$

$x_{i,j}$'s are sub-Gaussian random variables with parameter σ^2. Therefore,

$$\begin{aligned}\Pr\left[|\widetilde{\mu}_T^2 - \mu_T^2| > \tfrac{\epsilon m}{10|T|}\right] &= \Pr\left[|\widetilde{\mu}_T - \mu_T|\,(\widetilde{\mu}_T + \mu_T) > \tfrac{\epsilon mn}{10\,|T^I||T^J|}\right] \\ &\leq \Pr\left[|\widetilde{\mu}_T - \mu_T| > \tfrac{\epsilon mn}{20\,a\,|T^I||T^J|}\right] \\ &= \Pr\left[|\sum_{i \in T^I}\sum_{j \in T^J}\sum_{g=1}^{C}(x_{i,j,g} - \mu_{i,j})| > \tfrac{C\,\epsilon mn}{20\,a}\right] \\ &\leq 2\exp\left(-\tfrac{C\,\epsilon^2 m^2 n^2}{800a^2\,|T|\,\sigma^2}\right) \leq \tfrac{\delta}{2mm}\end{aligned}$$

By the union bound, the probability that one of the $2mn$ boundary segments has an inaccurate estimate is at most δ. □

2.3 The *ISplit-Grid* Algorithm

Given the claims in the previous sections, we now present our incrementally improving visualization generation algorithm for heatmaps *ISplit-Grid*. Given the parameters ϵ and δ, *ISplit-Grid* maintains the same guarantee of error (ϵ) in generating the segment approximations in each iteration. Theorem 8 and Lemma 5 suffice to show that the *ISplit-Grid* algorithm is a greedy approximator, that, at each iteration identifies a segment approximation that is at least ϵ close to the best segment approximation for that iteration.

Data: $X_a, X_b, Y, \delta, \epsilon$
Start with the 1-block approximator $M = (M_1)$.
for $k = 2, \ldots, m$ **do**
  $M_{k-1} = (\beta_1, \ldots, \beta_{k-1})$.
  for *each* $\beta_i \in M_{k-1}$ **do**
    for *each* *group combination* $x_{q,r} \in \beta_p$ **do**
      Draw C samples uniformly. Compute mean $\tilde{\mu}_{q,r}$
    end
    Compute $\widetilde{\mu}_{\beta_i} = \frac{1}{|\beta_i^I||\beta_i^J|} \sum_{q \in \beta_i^I} \sum_{r \in \beta_i^J} \tilde{\mu}_{q,r}$
  end
  Find $(T, U, V, W) = \text{argmax}_{p^*;\, T,U,V,W \subseteq \beta_{p^*}} \frac{|T^I||T^J|}{mn}\mu_T^2 + \frac{|U^I||U^J|}{mn}\mu_U^2 + \frac{|V^I||V^J|}{mn}\mu_V^2 + \frac{|W^I||W^J|}{mn}\mu_W^2 - \frac{|\beta_{p^*}^I| \times |\beta_{p^*}^J|}{mn}\mu_{\beta_{p^*}}^2$.
  Update $M_{k+1} = \beta_1, \ldots, \beta_{i-1}, T, U, V, W, \beta_{i+1}, \ldots, \beta_k$.
end

Algorithm 2: *ISplit-Grid*

3. RELEASING THE REFINEMENT RESTRICTION: *DPSplit*

In this section, we present an incremental visualization generation algorithm for trendlines, *DPSplit*. At each iteration k, *DPSplit* generates the entire k-segment approximation—releasing the refinement restriction. The algorithm works as follows: Given the task of approximating the distributions $D_1, \ldots, D_m$, at each iteration k, *DPSplit* draws samples uniformly from each group (as in *ISplit*) and then computes the ℓ_2-squared error of all possible k-segment approximations that can be generated. *DPSplit* then chooses the k-segment approximation that yields the least error to be output. Therefore, instead of refining the $(k-1)$-segment approximation, *DPSplit* computes the k-segment approximation based on the improved estimates of the means $\mu_1, \ldots, \mu_m$ of the distributions obtained at iteration k. We use the same notion of error of a segment $\text{err}(S_i)$ (Equation 5) as *ISplit*. We represent the error of a segment approximation that approximate the distributions $D_p, \ldots, D_q$ with k-segments as $\text{err}(L_{[p,q]}, k)$. For *ISplit*, $\text{err}(L_k) = \text{err}(L_{[1,m]}, k)$

Data: X_a, Y, δ, ϵ
Start with the 1-segment approximation: $\forall_{i \in [1,m]} D(i,1) = \text{err}(L_{[1,i]}, 1)$ and $P(i,1) = -1$. Set $L = (L_1)$
for $k = 2, \ldots, m$ **do**
  for $i = 1, \ldots, m$ **do**
    for $j = 2, \ldots, k$ *and* $j \leq i$ **do**
      $D(i,j) = \text{argmax}_{p^* \in [1, i-1]}\, D(p^*, j-1) + \text{err}(S([p^*+1, i], \eta))$
      $P(i,j) = p^*$ such that D(i,j) is minimized
    end
  end
  Recursively construct L_k by traversing P starting from $P(m,k)$
  $L = L \cup L_k$
end

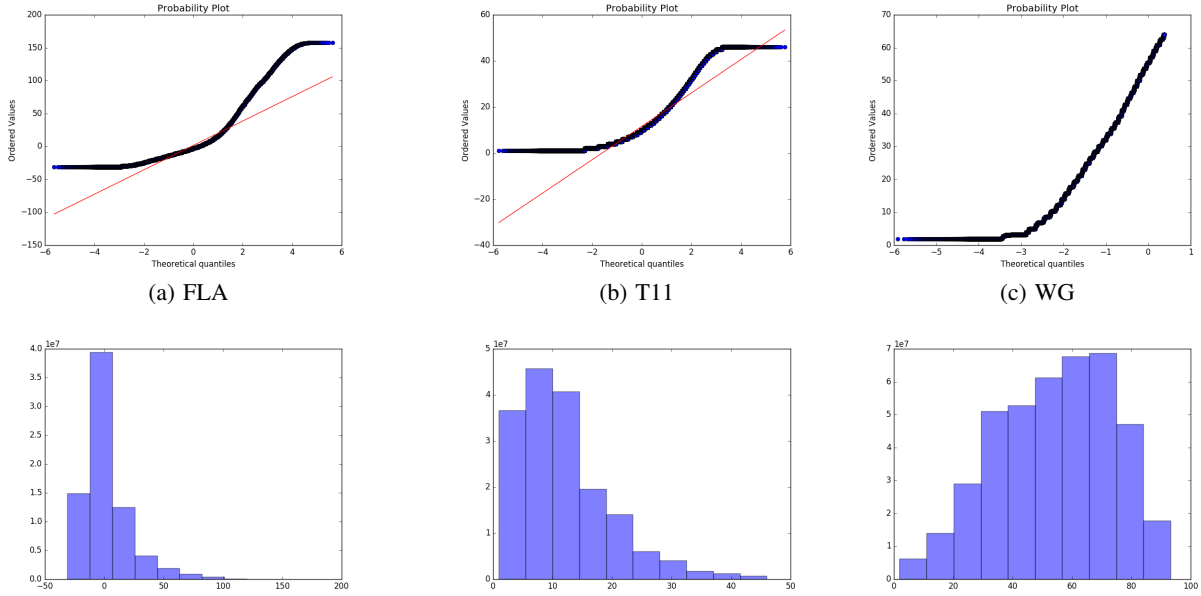
Algorithm 3: *DPSplit*

DPSplit maintains a $m \times k$ Dynamic Programming (DP) table D where each entry $D(i,j)$ corresponds to the error, $\text{err}(L_{[1,i]}, j)$ of the j-segment approximation of the distributions $D_1, \ldots, D_i$. $DPSplit$ also maintains another $m \times k$ table P, where each entry $P(i,j)$ corresponds to the *split group* that minimizes the error corresponding to $D(i,j)$. Given m distributions, at each iteration k, all the entries from $D(1,1)$ to $D(m,k)$ are updated, i.e., the error of the segment approximation is recomputed based on the new samples drawn from the distributions. In the final iteration $k = m$, the entire table is updated. Therefore, the time complexity of *DPSplit* is $\mathcal{O}(m \times k^2)$ even though the samples taken in each iteration is the same as *ISplit*.

4. DATASET ANALYSIS

In this section, we present the Q-Q plot [49] analysis of the datasets (Table 2) used in Section 5. We present the results for the *Arrival Delay* attribute of the Flight dataset (FLA), the *Trip Time* attribute of the NYC taxi dataset for the year 2011 (T11), and the *Temperature* attribute of the weather dataset (WG). We also present the corresponding histograms of the datasets. We exclude the results of the *Departure Delay* attribute of Flight dataset due to similarity in distribution to the *Arrival Delay* attribute. For the same reason, we also exclude the results for T12 and T13 (similarity to T11 dataset).

Figure 18(a) and 18(b) shows the Q-Q plot of FLA and T11, respectively. We plot the theoretical gaussian distribution in the x-axis and the ordered values of the attributes of the dataset in the y-axis. The shape of the plot determines the type of the distribution. Both the datasets exhibit a right skewed gaussian distribution confirmed by their corresponding histograms (Figure 18(d) and 18(e))—in both case, the peak is off center and the the tail stretches away from the center to the right side. On the other hand, the WG (Figure 18(c)) dataset exhibits a truncated gaussian distribution that is clipped on both sides (Figure 18(f)).



(a) FLA (b) T11 (c) WG

(d) FLA (e) T11 (f) WG

Figure 18: Q-Q plot of a) FLA, b) T11, and c) WG. Histogram of d) FLA, e) T11, and f) WG.