Robust Statistical Analysis on Streaming Data with Near-Duplicates in General Metric Spaces

QIN ZHANG, Indiana University, USA

This paper considers statistical analysis on noisy datasets where near-duplicate elements need to be treated as identical ones. We focus on two basic problems, distinct elements and ℓ_0-sampling, in the data stream model where the sequence of elements can only be scanned once using a limited space, under which a comprehensive data deduplication step before statistical analysis is not feasible. Previous streaming algorithms for these problems could only handle noisy datasets in $O(1)$-dimensional Euclidean spaces. In this paper, we propose sublinear-space streaming algorithms that work for noisy datasets in any metric space. We also give a lower bound result showing that solving the distinct elements problem on noisy datasets in general metric spaces is inherently more difficult than solving it on noiseless datasets and on noisy datasets in $O(1)$-dimensional Euclidean spaces.

CCS Concepts: • **Theory of computation → Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: data streams, near-duplicates, distinct elements, ℓ_0-sampling

ACM Reference Format:
Qin Zhang. 2025. Robust Statistical Analysis on Streaming Data with Near-Duplicates in General Metric Spaces. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 111 (May 2025), 25 pages. https://doi.org/10.1145/3725248

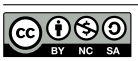
1 Introduction

Near-duplicates present a persistent challenge in the field of big data analytics. Conducting a comprehensive data cleaning step prior to the analytical phase may not always be feasible, especially in settings like the data stream model [4, 18], where available memory space is limited and cannot accommodate all the data elements. Previous studies [9, 10] have explored the feasibility of performing two basic statistical problems, distinct elements (or, F_0-estimation) and ℓ_0-sampling, in the data stream model without the need for a comprehensive data cleaning step. However, algorithms proposed in these works can only handle datasets in the $O(1)$-dimensional Euclidean space, which is *not* sufficient for many applications.

In this paper, we introduce streaming algorithms that can handle datasets with near-duplicates in general metric spaces while still using *sublinear* memory space. The bulk of this paper focuses on F_0-estimation and ℓ_0-sampling. In Section 7, we will briefly discuss several other statistical problems that can be solved in the noisy data setting by adapting algorithms designed for noiseless datasets.

Motivation. The original motivation for studying datasets with near-duplicates is that many real-world datasets are inherently noisy due to *human factors*. For example, the same mailing address is spelled differently in different contexts; the same image/video may appear in different forms due to compression, edits, and change of resolutions; the same query with different keywords combinations are sent to a search engine, etc. Such near-duplicates are ubiquitous in the age of big data and may lead to significant errors in subsequent data analytics if not appropriately managed.

Qin Zhang is partly supported by NSF CCF-1844234 and IU Luddy Faculty Fellowship.
Author's Contact Information: Qin Zhang, qzhangcs@iu.edu, Indiana University, Bloomington, IN, USA.



ACM 2836-6573/2025/5-ART111
https://doi.org/10.1145/3725248

We note that many of these data are *not* points in the Euclidean space. Although we can always embed them into vectors/points in the Euclidean space, the resulting dimension of these vectors can often be quite high.

Another strong motivation for studying datasets with near-duplicates comes from quantum data management, where the noise comes from *laws of nature*. In the quantum realm, it has been shown that $\Omega\left(d/\epsilon^2\right)$ copies of the quantum state are needed to learn a d-dimensional quantum state up to a trace distance ϵ [23, 33]. In other words, if we want to preserve quantum data classically[1] using a finite number of copies of the states, the resulting dataset is *inherently* noisy. Moreover, we need $d = 2^n$ dimensions to represent a n-qubit quantum state, which implies that the classical representation of quantum states are often vectors of very high dimensions, and thus cannot be handled by algorithms proposed in the previous work [9, 10].

As noticed in the previous work [9], it is unlikely that there is a magic hash function capable of hashing near-duplicates to the same element (while ensuring non-near-duplicates hash to different elements) with a description sublinear in terms of the input size. We thus cannot make use of existing streaming algorithms designed for noiseless datasets.

Robust F_0-Estimation and ℓ_0-Sampling. We now introduce the noisy data model and two statistical problems that we are going to study in this paper. We first give several definitions. Let $d(\cdot,\cdot)$ be a distance function.

Definition 1.1 (valid partition). Given a dataset Q, we call a group partition $\mathcal{G}(Q) = \{G_1, \ldots, G_n\}$ of Q *α-valid* if for any $i \in [n]$ and for any pair of elements $p, q \in G_i$, we have $d(p, q) \leq \alpha$.

In the noisy data setting, we consider elements in each group of a valid partition *near-duplicates*, and each group corresponds to the same ground truth element. The following definition for robust F_0 aligns with [9].

Definition 1.2 (robust F_0). The α-robust F_0 of a dataset Q, denoted by $F_0(Q, \alpha)$, is the number of groups in the minimum-cardinality α-valid partition $\mathcal{G}(Q) = \{G_1, \ldots, G_n\}$ of Q.

Note that the minimum-cardinality α-valid partition may not be unique, but this is not a problem since we are only interested in the minimum cardinality of such partitions.

W.l.o.g., we can assume that the distance threshold $\alpha = 1$, as we can always appropriately rescale the distance function. We will thus use "valid" instead of "α-valid", and write $F_0(Q, \alpha)$ as $F_0(Q)$ for simplicity. When Q is clear from the context, we will further simplify "$F_0(Q)$" and "$\mathcal{G}(Q)$" to "F_0" and "$\mathcal{G}$", respectively.

Definition 1.3 (well-shaped dataset). We say a dataset Q is *well-shaped* if it has a valid partition $\mathcal{G}(Q) = \{G_1, \ldots, G_n\}$ such that for any two elements $p, q \in Q$, where $p \in G_i$ and $q \in G_j$ $(j \neq i)$, we have $d(p, q) > 2$.

If such a partition exists, it must be unique and has the minimum cardinality. We call this partition the *natural partition* of the well-shaped dataset Q.

We define robust ℓ_0-sampling for well-shaped datasets as follows.

Definition 1.4 (robust ℓ_0-sampling). Let Q be a well-shaped dataset and $\mathcal{G} = \{G_1, \ldots, G_n\}$ be the natural group partition of Q. The robust ℓ_0-sampling on Q outputs an element $q \in Q$ with the property that for each $i \in [n]$, $\Pr[q \in G_i] = 1/n$.

We are also interested in datasets with small *F_0-ambiguity*, which is defined as follows:

[1]Even if large-scale quantum storage systems become available in the future, due to unique quantum properties including *post-measurement state disturbance* and *no-cloning theorem*, we need to store quantum data in a classical format within a database system to accommodate potentially unlimited queries.

Definition 1.5 (F_0-ambiguity). The F_0-ambiguity of a dataset Q is defined to be the minimum τ ($\tau \in [0, 1]$) such that there exists a set of outliers $O \subseteq Q$ satisfying the following: (1) $Q \backslash O$ is well-shaped, and (2) $F_0(Q \backslash O) \geq (1 - \tau)F_0(Q)$. We use $\tau(Q)$ to denote the F_0-ambiguity of Q.

The Data Stream Model. In the data stream model [4, 18], the elements $q_1, \ldots, q_m$ of the dataset come one by one in a sequence. During the streaming process, we need to maintain in the memory a sketch $sk(Q)$ of the set of elements Q that we have received so far, such that at any time step, we are able to output the value $f(Q)$ where $f(\cdot)$ is a function defined on Q (e.g., $f(Q) = F_0(Q)$ for the distinct elements problem) using $sk(Q)$. The primary goal in the data stream model is to minimize the sketch size.

Previous Approaches and Limitations. As mentioned, robust F_0-estimation and ℓ_0-sampling have been studied for points in the Euclidean space [9, 10]. The algorithms for F_0-estimation in [9] used the following idea: we first partition the Euclidean space using square grid such that each grid cell C intersects with *at most* one group in $\mathcal{G}$. Let $\mathcal{C}$ be the set of non-empty cells (i.e., cells containing at least one point in Q). We define the weight of a group G, denoted by $w(G)$, to be the number of cells $C \in \mathcal{C}$ that G intersects, and define the weight of each cell $C \in \mathcal{C}$ to be $w(C) = 1/w(G_C)$ if there is a (unique) group $G_C \in \mathcal{G}$ intersecting C, and $w(C) = 0$ otherwise. It is easy to see that $\sum_{C \in \mathcal{C}} w(C) = F_0$. For a $O(1)$-dimensional Euclidean space, we have $w(C) = \Theta(1)$. Therefore, by subsampling cells in $\mathcal{C}$ and then computing the sum of $w(C)$ for the sampled cells C, we can approximate the robust F_0.

To keep enough information in the memory for computing $w(C)$, for each incoming point q in the data stream, we first check whether there is a sampled cell C such that $d(q, C) \leq 1$. If yes, we store the cell C_q that contains q provided it has not been stored previously; otherwise, we discard q. Note that we are "preparing ahead". That is, we store q even if C_q is *not* sampled, as long as there is a possibility that another point in the same group falls into a sampled cell. Leveraging these stored information, we can calculate the weight of each sampled cell precisely.

Unfortunately, the above approach only works for low dimensional Euclidean spaces, as the number of cells that a group intersects may increase (or, the weight of each cell may decrease) *exponentially* as the dimension increases. We thus cannot guarantee that the memory usage is sublinear. More precisely, the space bound would be $\min\{2^{\Theta(d)}, m\}$, where d is the dimension of the Euclidean space and m is the length of the stream.

The algorithm for ℓ_0-sampling in [10] also uses the idea of grid partition. We again sample a subset of $\mathcal{C}$, but only consider groups whose *first* point is in a sampled cell as a candidate group for the final output. Again, this approach only works for low dimensional Euclidean spaces, because we also need to maintain groups whose first points are *not* in a sampled cell. This precaution is necessary to handle scenarios where subsequent points from these groups land in sampled cells, since we do *not* want to regard such groups as candidate groups. On the other hand, the number of cells that a group intersects may again increase exponentially as the dimension increases.

The works [9, 10] discussed how to handle high dimensional data and general distance functions. However, their algorithms only work for datasets exhibiting very high sparsity and for metric spaces admitting efficient locality sensitive hashing (LSH) schemes [25]. While in this paper, we aim to find solutions for general datasets in *any* metric spaces.

Our Contributions. The main contribution of this paper can be summarized as follows. For convenience, we assume that each element in the dataset can be stored in one word of memory space.[2] All algorithms succeed with probability 0.99.

[2]Otherwise, we need to multiply a factor of γ to all of our space upper bounds, where γ is the space needed to store one element in the data stream.

(1) For a well-shaped dataset, we give a streaming algorithm that computes a $(1+\epsilon)$-approximation of the robust F_0 using $O\left(\min\left\{\frac{1}{\epsilon}\sqrt{m\log\frac{1}{\epsilon}}, F_0\right\}\right)$ words of space (Section 2). The F_0 term is due to the fact that we can always store a representative element for each group. For simplicity, we will omit the trivial F_0 term in all of our upper bounds (or, assume that F_0 is larger than the other term in the min operation).
(2) We give a streaming algorithm for robust ℓ_0-sampling on a well-shaped dataset using $O(\sqrt{m})$ words of space. See Section 3.
(3) For a general dataset with F_0-ambiguity τ, we give a streaming algorithm that computes a $\frac{1+\epsilon}{1-\tau}$-approximation of robust F_0 and uses $O\left(\frac{1}{\epsilon}\sqrt{m\log m}\right)$ words of space. See Section 4.
(4) We show that any streaming algorithm for approximating robust F_0 up to a multiplicative factor 1.1 needs to use at least $\Omega(\sqrt{m})$ bits of space. See Section 5.

Our results indicate that the robust F_0-estimation problem has a significantly higher complexity than the F_0-estimation problem on noiseless datasets. For the latter, it is known that $O\left(\log m + 1/\epsilon^2\right)$ bits of space is sufficient for obtaining a $(1+\epsilon)$-approximation in the data stream model [27].

In the previous work [9, 10], it has been shown that for well-shaped datasets in the $O(1)$-dimensional Euclidean space, there exist streaming algorithms that compute $(1+\epsilon)$-approximations of robust F_0 (or, perform robust ℓ_0-sampling) with probability 0.99 using space $O\left(1/\epsilon^2\right)$ words (or, $O(1)$ words). Our results indicate that in the data stream model, solving the two problems on datasets in general metric spaces is significantly more difficult than that in the $O(1)$-dimensional Euclidean space.

We also obtain the following byproducts:

(5) For elements being points in the $O(1)$-dimensional Euclidean space, our algorithms designed for the general metric space can be modified to compute a $(1+\epsilon)$-approximation of the robust F_0 with probability 0.99 using $O\left(1/\epsilon^2\right)$ words of space, and perform ℓ_0-sampling with probability 0.99 using O(1) words of space.

These results match the performance of algorithms in [9, 10].[3]

Related Work. The study of statistical analysis on stream data has a long history, starting from [4, 18]. The F_0-estimation and ℓ_0-sampling are two of the most extensively studied problems. The first streaming algorithm for F_0-estimation was designed by Flajolet and Martin [18] in 1984. After several decades of research (e.g., [4, 7, 8, 15, 17, 20]), optimal space complexity has been obtained in [27]. Among many other applications in network traffic monitoring, data integration, and data warehousing, F_0-estimation serves as a foundation for database query plan optimization.

The streaming ℓ_0-sampling problem was studied in [13, 19, 22, 26], where [26] obtained the optimal space bound. This problem serves as a building block in numerous graph and geometry problems in the data stream model [1–3, 5, 11, 11, 12, 19, 29].

On the other hand, the literature on robust statistical estimation is rather sparse. As mentioned, [9, 10] studied F_0-estimation and ℓ_0-sampling, but only for data points in $O(1)$-dimensional Euclidean spaces or for datasets exhibiting high sparsity in metric spaces that support efficient LSH schemes. [36] studied statistical problems for datasets with near-duplicates in the distributed computation model, where the communication cost is the primary concern.

Finally, we would like to mention that there is a large literature on data deduplication (also called *entity resolution* or *record linkage*) [14, 16, 24, 30]. However, as previously mentioned, we cannot afford a comprehensive data deduplication step in the data stream model.

[3]The algorithm for robust ℓ_0-sampling in [10] uses $O(\log m)$ space with a success probability $(1-1/m)$. By standard parallel repetition, this is equivalent to an algorithm that achieves a success probability 0.99 using $O(1)$ space.

Roadmap. The rest of this paper is organized as follows. After introducing useful notations and conventions (Section 1.1), we present our algorithms for approximating robust F_0 of a well-shaped dataset in Section 2. We then make use of our algorithm for robust F_0-estimation to design an algorithm for robust ℓ_0-sampling in Section 3. In Section 4, we give an alternative algorithm for ℓ_0-sampling on well-shaped datasets, and then show that it can be used to approximate robust F_0 on general datasets. We provide our lower bound result in Section 5, and conclude the paper in Section 8.

1.1 Preliminaries

Notations. We will use the following notations in this paper. All logarithms are in base 2 unless specified otherwise. Let $[n] \triangleq \{1, \ldots, n\}$.

- m: the length of the data stream.
- U: the universe of elements.
- Q ($\subseteq U$): the set of m elements in the data stream.
- $\mathcal{G} \triangleq \mathcal{G}(Q)$: the set of groups in the minimum-cardinality partition of Q. When Q is well-shaped, $\mathcal{G}$ is the natural partition.
- q_G: the representative element of group G; its selection varies in different algorithms.
- t_G: the number of elements of group G that arrive after q_G (including q_G) in the data stream.
- $\mathcal{G}^{\text{sample}}$: the set of groups $G \in \mathcal{G}$ in which at least one element has been sampled.

We note that during the streaming process, q_G, t_G, and $\mathcal{G}^{\text{sample}}$ may change over time. In all of our algorithms, each group $G \in \mathcal{G}^{\text{sample}}$ is represented by a tuple containing $O(1)$ values.

The Zero$(\cdot)$ Function and Global Sampling Threshold z. We begin with h as a random hash function $h_0 : U \to \{0,1\}$. When m doubles for the i-th time, we choose a new random hash function $h_i : U \to \{0,1\}$, and append the output of h to the output of h_i. That is, for an element $q \in U$, $h(q) = h_i(q) \circ h_{i-1}(q) \circ \ldots \circ h_0(q)$, where "$\circ$" denotes concatenation. For an element $q \in U$, let $\texttt{zero}(q)$ be the number of trailing zeros of $h(q)$.

In all of our algorithms, we maintain a global sampling threshold z. For each element $q \in Q$, we sample q if $\texttt{zero}(q) \geq z$; equivalently, we sample q with probability $1/2^z$.

Conventions. For any $\epsilon \in [0,1]$, we say $\tilde{x}$ is a $(1+\epsilon)$-approximation of x if $(1-\epsilon)x \leq \tilde{x} \leq (1+\epsilon)x$. We often ignore floor and ceiling operations when they do not affect any asymptotic bounds.

We will use constants such as 0.001 and 0.999 to represent low probability and high probability, respectively. They can be replaced by δ and $1-\delta$ for any $\delta > 0$ by the standard parallel repetition, at the cost of an extra $\log(1/\delta)$ factor in the space cost.

For simplicity, we will use fully random hash functions, which can always be replaced by pseudo-random hash functions by Nisan's pseudorandom generator [32] for space-bounded computation at a small (logarithmic in m) extra space cost and an additional $1/m$ error probability, which will not affect the asymptotic performance of our algorithms.

Probability Tools. We will make use of the following version of the Chernoff bound.

Lemma 1.6. *Let $X_1, \ldots, X_n$ be independent random variables such that $\forall i \in [n], X_i \in [0,1]$. Let $X = \sum_{i\in[n]} X_i$ and let $\mu = \mathbf{E}[X]$. For any $\delta \in (0,1)$, it holds that $\Pr[|X-\mu| \geq \delta\mu] \leq 2e^{-\delta^2\mu/3}$. And for any $a > 0$, we have $\Pr[X > a] \leq e^{-(a-2\mu)}$.*

2 Robust F_0 for Well-Shaped Datasets

In this section, we present an algorithm for robust F_0-estimation on well-shaped datasets.

The Ideas. We begin by giving an overview of the ideas behind our algorithm. The naive algorithm to compute robust F_0 involves maintaining one representative element from each group, which clearly requires $\Theta(F_0)$ space. A natural approach to reduce space usage is to sample groups by sampling the stream elements. However, this subsampling procedure will cause large groups to be sampled with higher probability than small groups. If we have information about the sizes of the groups, we could compute a weight for each group to ensure that each group is treated equally. The challenge is that by the time we encounter a sampled element $q_G \in G$, we may have already discarded many elements in G that came before q_G, making it difficult to accurately determine the size of G.

The algorithm for datasets in the $O(1)$-dimensional Euclidean space [9] resolves this issue by sampling grid cells (instead of elements in Q), which can be done independent of the data stream. As mentioned previously, this method comes at a price: it is tailored for the Euclidean space and the sketch size grows exponentially with the space dimension.

We take a different approach to circumvent the above issue. For each sampled group $G \in \mathcal{G}^{\text{sample}}$, we count the number of elements that arrive after a *randomly* sampled element q_G; we denote this count by t_G. We introduce the following quantities and notations to facilitate the further discussion.

- k: a threshold parameter proportional to $\sqrt{m}$.
- $\mathcal{G}^{\text{large}}$: the set of groups $G \in \mathcal{G}^{\text{sample}}$ with $t_G > k$.
- $\mathcal{G}^{\text{small}} \triangleq \mathcal{G} \backslash \mathcal{G}^{\text{large}}$. Note that $\mathcal{G}^{\text{small}}$ is the *union* of the set of groups $\{G \in \mathcal{G}^{\text{sample}} \mid t_G \le k\}$ and the set of groups $\mathcal{G} \backslash \mathcal{G}^{\text{sample}}$.
- n_j $(j = 1, \ldots, k-1)$: the number of groups $G \in \mathcal{G}^{\text{small}}$ with $|G| = j$.
- n_k: the number of groups $G \in \mathcal{G}^{\text{small}}$ with $|G| \ge k$. Note that n_k is defined differently from n_j $(j = 1, \ldots, k-1)$.
- X_j $(j \in [m])$: the number of groups $G \in \mathcal{G}^{\text{sample}}$ with $t_G = j$. Note that X_j can be computed at any time in the streaming process if we are able to store t_G for each $G \in \mathcal{G}^{\text{sample}}$.

We manage to establish the following connections between the tail length frequencies $\{X_j\}$ of sampled groups in $\mathcal{G}^{\text{sample}}$ and the group size frequencies $\{n_j\}$ over all groups in $\mathcal{G}^{\text{small}}$ when we sample each element in the stream with probability $1/k$:

(1) For any $j < k$, the quantity $k(\mathbf{E}[X_j] - \mathbf{E}[X_{j+1}])$ is roughly equal to n_j.
(2) The quantity $k\mathbf{E}[X_k]$ is roughly equal to n_k.

On the other hand, it is clear that $\sum_{j=k+1}^{m} X_j$ is exactly $\left|\mathcal{G}^{\text{large}}\right|$. Therefore, we can use observations X_j $(j = 1, \ldots, m)$ to estimate $F_0 = \left|\mathcal{G}^{\text{large}}\right| + \left|\mathcal{G}^{\text{small}}\right| = \sum_{j=k+1}^{m} X_j + \sum_{j=1}^{k} n_j \approx \sum_{j=k+1}^{m} X_j + \sum_{j=1}^{k-1} k(\mathbf{E}[X_j] - \mathbf{E}[X_{j+1}]) + k\mathbf{E}[X_k]$, which is roughly what Algorithm 1 returns at the time of query.

Algorithm and Analysis. Our algorithm is presented in Algorithm 1. We have the following result.

Theorem 2.1. *Algorithm 1 outputs a $(1+\epsilon)$-approximation of the robust F_0 of a well-shaped dataset of m elements with probability 0.99 using $O\left(\frac{1}{\epsilon}\sqrt{m \log \frac{1}{\epsilon}}\right)$ words of space.*

In the rest of this section, we prove Theorem 2.1.

Since we sample each element q in the stream if $\texttt{zero}(q) \ge z$ where $z = \log k$,[4] we are essentially sampling each element with probability $1/k$, where the value of k is chosen at Line 3 of Algorithm 1. We thus have $\mathbf{E}\left[\left|\mathcal{G}^{\text{sample}}\right|\right] \le \frac{m}{k}$. By a Chernoff bound,

$$\Pr\left[\left|\mathcal{G}^{\text{sample}}\right| - \mathbf{E}\left[\left|\mathcal{G}^{\text{sample}}\right|\right] > \frac{m}{2k}\right] \le \exp\left(-\frac{m}{12k}\right) \le 0.001.$$

[4]We ignore the floor/ceiling operations and assume k is a power of 2; this will not change any asymptotic bounds.

Algorithm 1: ROBUST-F_0-ESTIMATION

Input: a stream of well-shaped dataset Q, parameter ϵ
Output: a $(1+\epsilon)$-approximation of $F_0(Q)$

$\varepsilon \leftarrow \epsilon/5$, $m \leftarrow 1$, $\mathcal{G}^{\text{sample}} \leftarrow \emptyset$, $z \leftarrow 0$
foreach *element q in the stream* **do**
  $m \leftarrow m+1$, $k \leftarrow \max\left\{\sqrt{\frac{\varepsilon^2 m}{100\log\frac{1}{\varepsilon}}}, 1\right\}$, $z \leftarrow \lfloor \log k \rfloor$
  foreach $G \in \mathcal{G}^{\text{sample}}$ **do**
    /* if representative element q_G is no longer sampled, delete G */
    if $\texttt{zero}(q_G) < z$ **then** delete G from $\mathcal{G}^{\text{sample}}$
  if $\exists G \in \mathcal{G}^{\text{sample}}$ *s.t.* $d(q_G, q) \le 1$ **then**
    /* if q belongs to some group G, try to maintain the representative element q_G as a random element from those elements in G with the highest $\texttt{zero}(\cdot)$ value */
    if $\texttt{zero}(q) > \texttt{zero}(q_G)$ **then**
      $q_G \leftarrow q$, $r_G \leftarrow 1$, $t_G \leftarrow 1$ /* whenever q_G gets updated, we reset t_G */
    else if $\texttt{zero}(q) = \texttt{zero}(q_G)$ **then**
      /* In the case of ties, use Reservoir sampling [34] among elements with the highest $\texttt{zero}(\cdot)$ value; r_G is a counter used in Reservoir sampling */
      $r_G \leftarrow r_G + 1$, $t_G \leftarrow 1$
      w.pr. $\frac{1}{r_G}$, $q_G \leftarrow q$
    else $t_G \leftarrow t_G + 1$
  else if $\texttt{zero}(q) \ge z$ **then**
    /* for a newly sampled element that doesn't belong to any existing group, create a new group $G = (q_G, t_G, r_G)$ and add it to $\mathcal{G}^{\text{sample}}$; we initialize the representative element q_G as q, the tail length t_G to be 1, and the counter r_G to be 1 */
    create a tuple $G \triangleq (q_G, t_G, r_G) \leftarrow (q, 1, 1)$ and add G to $\mathcal{G}^{\text{sample}}$
/* at the time of query */
for $j = 1, \ldots, m$ **do**
  Let X_j be the number of groups $G \in \mathcal{G}^{\text{sample}}$ with $t_G = j$
Let $\ell_1, \ldots, \ell_B$ and $\beta_1, \ldots, \beta_B$ be the pre-computed values defined in Definition 2.2
return $\tilde{F}_0 = \sum_{j=k+1}^{m} X_j + \sum_{b\in[B-1]} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B}$

This implies that with probability 0.999, we have $\left|\mathcal{G}^{\text{sample}}\right| \le \mathbf{E}\left[\left|\mathcal{G}^{\text{sample}}\right|\right] + \frac{m}{2k} \le \frac{3m}{2k} < \frac{1}{\varepsilon}\sqrt{400m\log\frac{1}{\varepsilon}}$. Therefore, the space usage of Algorithm 1 is bounded by $M \triangleq \frac{1}{\varepsilon}\sqrt{400m\log\frac{1}{\varepsilon}}$.

We assume that $F_0 \ge M$, since otherwise we can store all groups in $\mathcal{G}$. We then have

$$\frac{F_0}{k} \ge \frac{M}{k} = \frac{1}{\varepsilon}\sqrt{400m\log\frac{1}{\varepsilon}} \Bigg/ \sqrt{\frac{\varepsilon^2 m}{100\log(1/\varepsilon)}} \ge \frac{100}{\varepsilon^2}\log\frac{1}{\varepsilon}. \tag{1}$$

Recall the sets $\mathcal{G}^{\text{large}}, \mathcal{G}^{\text{small}}$ and values X_j ($j \in [m]$), n_j ($j \in [k]$) defined previously. By the definitions of $\mathcal{G}^{\text{large}}$ and X_j, we have

$$T \triangleq \sum_{j=k+1}^{m} X_j = \left|\mathcal{G}^{\text{large}}\right|. \tag{2}$$

Since $F_0 = |\mathcal{G}| = \left|\mathcal{G}^{\text{large}}\right| + \left|\mathcal{G}^{\text{small}}\right|$, the remaining task is to estimate $\left|\mathcal{G}^{\text{small}}\right|$ using $X_1, \ldots, X_k$.

The Expectation of X_j. We first investigate the expectation of each X_j. For each $i \in [k]$, let $p_i = 1 - (1 - 1/k)^i$ be the probability that there is at least one sampled element in a group G with $|G| = i$. And let

$$\lambda_i \triangleq \frac{p_i}{i} = \frac{1 - \left(1 - \frac{1}{k}\right)^i}{i}. \tag{3}$$

It is not difficult to check that

$$\lambda_1 > \lambda_2 > \ldots > \lambda_k. \tag{4}$$

Direct calculation gives

$$\forall i \in [k], \quad \lambda_i \in [\lambda_k, \lambda_1] \subseteq \left[\frac{1 - e^{-1}}{k}, \frac{1}{k}\right]. \tag{5}$$

For each $G \in \mathcal{G}^{\text{small}}$ and $j \in [k]$, let $Y_{G,j} \in \{0, 1\}$ be the random variable such that

$$Y_{G,j} = \begin{cases} 1, & \text{if } G \in \mathcal{G}^{\text{sample}} \text{ and } t_G = j, \\ 0, & \text{otherwise.} \end{cases}$$

We thus have

$$X_j = \sum\nolimits_{G \in \mathcal{G}^{\text{small}}} Y_{G,j}. \tag{6}$$

Note that conditioned on $G \in \mathcal{G}^{\text{small}} \cap \mathcal{G}^{\text{sample}}$, by Reservoir sampling (Line 14 of Algorithm 1), q_G is a random element among the last $k_G \triangleq \min\{k, |G|\}$ elements in G. We thus have

$$\Pr\left[Y_{G,j} = 1\right] = p_{k_G} \cdot \frac{1}{k_G} = \lambda_{k_G}. \tag{7}$$

By (6) and (7), we have the following relation between X_j and n_j ($j \in [k]$).

$$\begin{aligned} \mathbf{E}[X_j] &= \sum_{G \in \mathcal{G}^{\text{small}}} \mathbf{E}[Y_{G,j}] = \sum_{i=j}^{k-1} \sum_{G \in \mathcal{G}^{\text{small}}: |G| = i} \lambda_i + \sum_{G \in \mathcal{G}^{\text{small}}: |G| \geq k} \lambda_k \\ &= \lambda_j n_j + \ldots + \lambda_{k-1} n_{k-1} + \lambda_k n_k. \end{aligned} \tag{8}$$

We observe the following properties of $\mathbf{E}[X_j]$:

(1) $\mathbf{E}[X_j]$ ($j = 1, \ldots, k$) monotonically decrease, and

$$\mathbf{E}[X_j] \leq \lambda_1 \sum_{i=j}^{k} n_i \leq \lambda_1 F_0. \tag{9}$$

(2) We can obtain n_j ($j \in [k]$) from $\mathbf{E}[X_j]$'s as follows:

$$\forall 1 \leq j < k, n_j = \frac{\mathbf{E}[X_j] - \mathbf{E}[X_{j+1}]}{\lambda_j}; \quad n_k = \frac{\mathbf{E}[X_k]}{\lambda_k}. \tag{10}$$

The Estimator. Equation (10) suggests us to estimate $\left|\mathcal{G}^{\text{small}}\right|$ using following expression:

$$\sum_{j \in [k-1]} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_k}{\lambda_k}. \tag{11}$$

To facilitate the analysis, we try to reduce the number of summands in (11) and modify the estimator accordingly.

Definition 2.2 (B, β_b, ℓ_b). Let

$$B = 1 + \log_{1+\varepsilon} \frac{\lambda_1}{\lambda_k} \in \left(\frac{1}{2\varepsilon}, \frac{1}{\varepsilon}\right). \tag{12}$$

We define

- $\beta_1 = \lambda_1$; $\beta_b = \lambda_1 \big/ (1+\varepsilon)^{b-1}$ $(b = 2, \ldots, B)$; we thus have $\beta_B = \lambda_k$, and $\beta_1 > \beta_2 > \ldots > \beta_B$.
- $\ell_1 = 1$; let ℓ_b $(b = 2, \ldots, B)$ be the smallest index such that $\lambda_{\ell_b} \le \beta_b$; we thus have $\ell_B = k$.

We define the estimator of $\left|\mathcal{G}^{\text{small}}\right|$ as

$$S \triangleq \sum_{b\in[B-1]} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B}. \tag{13}$$

The final estimator for $|\mathcal{G}|$ can be written as

$$\tilde{F}_0 = S + T = \sum_{j=k+1}^{m} X_j + \sum_{b\in[B-1]} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B}. \tag{14}$$

We have the following lemma.

Lemma 2.3. *With probability at least* 0.99, $\left|S - \left|\mathcal{G}^{\text{small}}\right|\right| \le 5\varepsilon F_0$.

Proof. By (8), we have

$$\mathbf{E}[X_{\ell_B}] = n_{\ell_B}\beta_B = n_k\beta_B,$$

where the last equality follows from the definition $\ell_B = k$. And for any $b \in [B-1]$,

$$\begin{aligned}\mathbf{E}[X_{\ell_b}] - \mathbf{E}[X_{\ell_{b+1}}] &= \sum_{j\in[\ell_b,\ell_{b+1})} \lambda_j n_j \\ &\in \left[\frac{\beta_b}{1+\varepsilon} \sum_{j\in[\ell_b,\ell_{b+1})} n_j,\ \beta_b \sum_{j\in[\ell_b,\ell_{b+1})} n_j\right].\end{aligned}$$

Therefore, the quantity

$$\mathbf{E}[S] = \sum_{b\in[B-1]} \frac{\mathbf{E}[X_{\ell_b}] - \mathbf{E}[X_{\ell_{b+1}}]}{\beta_b} + \frac{\mathbf{E}[X_{\ell_B}]}{\beta_B}$$

estimates $\left|\mathcal{G}^{\text{small}}\right| = \sum_{j\in[k]} n_j = \sum_{b\in[B-1]} \sum_{j\in[\ell_b,\ell_{b+1})} n_j + n_k$ up to a multiplicative factor $(1+\varepsilon)$. Consequently,

$$\left|\mathbf{E}[S] - \left|\mathcal{G}^{\text{small}}\right|\right| \le \varepsilon\left|\mathcal{G}^{\text{small}}\right| \le \varepsilon F_0. \tag{15}$$

We next bound the difference between S and $\mathbf{E}[S]$. To facilitate the analysis, we first split the sum in (13) into two parts. Let b^* be the index such that $\mathbf{E}[X_{\ell_{b^*}}] > \varepsilon\lambda_k F_0$ and $\mathbf{E}[X_{\ell_{b^*}+1}] \le \varepsilon\lambda_k F_0$. We assume the existence of such a b^*; the case where no such b^* exists will be addressed at the end of the proof. We write

$$S = U + V, \tag{16}$$

where

$$U = \sum_{b\in[b^*-1]} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_{b^*}}}{\beta_{b^*}},$$

and

$$V = -\frac{X_{\ell_{b^*+1}}}{\beta_{b^*}} + \sum_{b=b^*+1,\ldots,B-1} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B}.$$

We first analyze U:

$$\begin{aligned}
\mathbf{E}[U] &= \sum_{b\in[b^*-1]} \frac{\mathbf{E}[X_{\ell_b}] - \mathbf{E}[X_{\ell_{b+1}}]}{\beta_b} + \frac{\mathbf{E}[X_{\ell_{b^*}}]}{\beta_{b^*}} \\
&= \sum_{b\in[b^*-1]} \left(\frac{1}{\beta_{b+1}} - \frac{1}{\beta_b}\right) \mathbf{E}[X_{\ell_{b+1}}] + \frac{\mathbf{E}[X_{\ell_1}]}{\beta_1} \\
&\le \sum_{b\in[b^*-1]} \frac{\varepsilon \mathbf{E}[X_{\ell_{b+1}}]}{\beta_{b+1}} + \frac{\mathbf{E}[X_{\ell_1}]}{\beta_1}. \qquad (17)
\end{aligned}$$

We bound the difference between X_{ℓ_b} and $\mathbf{E}[X_{\ell_b}]$ for each $b \in \{1,\ldots,b^*\}$. Set $\eta \triangleq \frac{\lambda_k F_0}{2B}$. By the definition of b^*, we know that $\mathbf{E}[X_j] \ge \varepsilon\lambda_k F_0$ for any $j \in [\ell_{b^*}]$. Therefore, for any $b \in [b^*-1]$,

$$\frac{\eta}{\mathbf{E}[X_{\ell_b}]} \le \frac{\lambda_k F_0}{2B \cdot \varepsilon\lambda_k F_0} = \frac{1}{2\varepsilon B} < 1.$$

On the other hand, by (9) we have $\mathbf{E}[X_{\ell_b}] \le \lambda_1 F_0$ for any $j \in [k]$. Therefore,

$$\frac{\eta}{\mathbf{E}[X_{\ell_b}]} \ge \frac{\lambda_k F_0}{2B \cdot \lambda_1 F_0} \ge \frac{1}{4B}.$$

By a Chernoff bound, we have for any $b \in [b^*-1]$,

$$\begin{aligned}
\Pr\left[\left|X_{\ell_b} - \mathbf{E}[X_{\ell_b}]\right| \ge \eta\right] &\le 2\exp\left(-\frac{\eta^2}{3\mathbf{E}[X_{\ell_b}]}\right) \\
&\le 2\exp\left(-\frac{\lambda_k F_0}{24B^2}\right) \qquad (18) \\
&\le 2\exp\left(-\frac{\varepsilon^2 F_0}{48k}\right) \qquad (19) \\
&\le 2\exp\left(-\log B^2\right) \qquad (20) \\
&\le \frac{10^{-3}}{B}, \qquad (21)
\end{aligned}$$

where from (18) to (19) we have used (5) and (12); from (19) to (20) we have used (1) and (12).

By another Chernoff bound, we have

$$\begin{aligned}
\Pr\left[\left|X_{\ell_1} - \mathbf{E}[X_{\ell_1}]\right| \ge \frac{\varepsilon\lambda_k F_0}{2}\right] &\le 2\exp\left(-\frac{\varepsilon^2\lambda_k^2 F_0^2}{12\lambda_1 F_0}\right) \qquad (22) \\
&\le 2\exp\left(-\frac{\varepsilon^2 F_0}{48k}\right) \qquad (23) \\
&\le \frac{10^{-3}}{B}, \qquad (24)
\end{aligned}$$

where from (22) to (23) we have used (5), and from (23) to (24) we have used (1) and (12).

By (21), (24), and a union bound, we have with probability 0.998,

$$\begin{aligned} |U - \mathbf{E}[U]| &\leq \sum_{b\in[b^*-1]} \frac{\varepsilon}{\beta_{b+1}} \cdot \eta + \frac{\varepsilon\lambda_k F_0}{2\beta_k} \\ &\leq B \cdot \frac{\varepsilon}{\beta_{b+1}} \cdot \frac{\lambda_k F_0}{2B} + \frac{\varepsilon\lambda_k F_0}{2\beta_k} \\ &\leq \frac{\varepsilon F_0}{2} + \frac{\varepsilon F_0}{2} = \varepsilon F_0. \end{aligned} \tag{25}$$

We next bound V. It is easy to see that

$$\begin{aligned} 0 \leq V &\leq \sum_{b=b^*+1,\ldots,B-1} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B} \\ &\leq \frac{1}{\beta_B}\left(\sum_{b=b^*+1,\ldots,B-1} (X_{\ell_b} - X_{\ell_{b+1}}) + X_{\ell_B}\right) \\ &= \frac{1}{\lambda_k} \cdot X_{\ell_{b^*+1}}. \end{aligned} \tag{26}$$

Recall that by the definition of b^*, we have $\mathbf{E}[X_{\ell_{b^*+1}}] \leq \varepsilon\lambda_k F_0$. By Lemma 1.6, it holds that

$$\Pr\left[X_{\ell_{b^*+1}} > 3\varepsilon\lambda_k F_0\right] \leq \exp(-(3\varepsilon\lambda_k F_0 - 2\varepsilon\lambda_k F_0)) \leq 0.001. \tag{27}$$

Combining (26) and (27), we have that with probability 0.999,

$$0 \leq V \leq 3\varepsilon F_0. \tag{28}$$

By (16), (25) and (28), we have that with probability 0.997,

$$|S - \mathbf{E}[S]| \leq 4\varepsilon F_0. \tag{29}$$

Lemma 2.3 follows from (15) and (29).

Finally, in the case when no such b^* exists, we write

$$S = \sum_{b=1,\ldots,B-1} \frac{X_{\ell_b} - X_{\ell_{b+1}}}{\beta_b} + \frac{X_{\ell_B}}{\beta_B}.$$

By the same argument as that for V, we have that with probability 0.999, $0 \leq S \leq 3\varepsilon F_0$. We thus have $|S - \mathbf{E}[S]| \leq 3\epsilon F_0$, combining which with (15), the lemma follows. □

By (2), (14), Lemma 2.3 and the fact that $F_0 = \left|\mathcal{G}^{\text{small}}\right| + \left|\mathcal{G}^{\text{large}}\right|$, we have with probability 0.99,

$$\left|\tilde{F}_0 - F_0\right| = \left|S - \left|\mathcal{G}^{\text{small}}\right|\right| \leq 5\varepsilon F_0.$$

Setting $\epsilon = \varepsilon/5$, we have that $\tilde{F}_0$ is a $(1+\epsilon)$-approximation of F_0.

3 Robust ℓ_0-Sampling for Well-Shaped Datasets

In this section, we give an algorithm for robust ℓ_0-sampling using our algorithm for F_0-estimation.

The idea of the ℓ_0-sampling algorithm is to perform a rejection sampling on the *last* element q_G^{last} of each group $G \in \mathcal{G}^{\text{sample}}$ with probability $p_z = \frac{1}{2^z}$ at the time of query. This is different from the approach in the previous work [10] for robust ℓ_0-sampling, where we try to store all groups which are close to the set of sampled cells and then check whether their *first* points are in sampled cells. To make sure that we still have "surviving groups" after the rejection sampling step *and* the space cost is sublinear during the entire streaming process, we have to carefully choose the sample

Algorithm 2: ROBUST-ℓ_0-SAMPLING

Input: a stream of well-shaped dataset Q
Output: an element of a group randomly sampled from $\mathcal{G}(Q)$

1 $z \leftarrow 0, \mathcal{G}^{\text{sample}} \leftarrow \emptyset, \mathcal{G}^{\text{accept}} \leftarrow \emptyset$
2 **foreach** *element q in the stream* **do**
3 /* maintain an estimate of F_0 for continuously updating the global sample rate */
4 $\tilde{F}_0 \leftarrow$ ROBUST-F_0-ESTIMATION$(q, 0.1)$, $z \leftarrow \lfloor \log \frac{\tilde{F}_0}{10} \rfloor$
5 **foreach** $G \in \mathcal{G}^{\text{sample}}$ **do**
6 /* if representative element q_G is no longer sampled, delete G */
7 **if** $\texttt{zero}(q_G) < z$ **then** delete G from $\mathcal{G}^{\text{sample}}$
8 **if** $\exists G \in \mathcal{G}^{\text{sample}}$ *s.t.* $d(q_G, q) \le 1$ **then**
9 /* if q belongs to some group G, update q_G^{last} */
10 $q_G^{\text{last}} \leftarrow q$
11 /* update q_G to be q if necessary */
12 **if** $\texttt{zero}(q) > \texttt{zero}(q_G)$ **then** $q_G \leftarrow q$
13 **else**
14 /* for a newly sampled element that doesn't belong to any existing group, create a new group $G = (q_G, q_G^{\text{last}})$ and add it to $\mathcal{G}^{\text{sample}}$; we initialize both the representative element q_G and q_G^{last} as q */
15 **if** $\texttt{zero}(q) \ge z$ **then**
16 create a tuple $G \triangleq (q_G, q_G^{\text{last}}) \leftarrow (q, q)$ and add G to $\mathcal{G}^{\text{sample}}$
17 /* at the time of query: */
18 **for** $G \in \mathcal{G}^{\text{sample}}$ **do**
19 /* perform rejection sampling on the last element of all groups in $\mathcal{G}^{\text{sample}}$ at the time of query */
20 **if** $\texttt{zero}(q_G^{\text{last}}) \ge z$ **then** add q_G^{last} to $\mathcal{G}^{\text{accept}}$
21 **if** $\left|\mathcal{G}^{\text{accept}}\right| = 0$ **then return** "failure"
22 **else return** a random element in $\mathcal{G}^{\text{accept}}$

rate p_z. We show that with the help of an estimation of the number of distinct elements F_0, we can maintain an appropriate p_z for our purposes during the streaming process.

Our algorithm is presented in Algorithm 2. We have the following result.

THEOREM 3.1. *Algorithm 2 solves robust ℓ_0-sampling on a well-shaped dataset of m elements using $O(\sqrt{m})$ words of space; the algorithm succeeds with probability* 0.99.

PROOF. For the correctness, it is easy to see that all groups $G \in \mathcal{G}$ with $\texttt{zero}(q_G^{\text{last}}) \ge z$ are included in $\mathcal{G}^{\text{sample}}$. By the construction of $\mathcal{G}^{\text{accept}}$ (Line 18-20 in Algorithm 2), it includes all (and only) groups $G \in \mathcal{G}$ with $\texttt{zero}(q_G^{\text{last}}) \ge z$. Therefore, we just need to make sure that $\left|\mathcal{G}^{\text{accept}}\right| \ge 1$ at the end of the streaming process, the probability of which is equal to $(1 - p_z)^{F_0}$. We thus set

$p_z = \frac{10}{\tilde{F}_0}$, where $\tilde{F}_0$ is a 1.1-approximation of F_0. Under this setting, we have

$$(1-p_z)^{F_0} = \left(1-\frac{10}{\tilde{F}_0}\right)^{F_0} \le \left(1-\frac{9}{F_0}\right)^{F_0} \le 0.001. \tag{30}$$

We can maintain $\tilde{F}_0$ in parallel using our algorithm for robust F_0-estimation (Algorithm 1).

By a Chernoff bound, the number of sampled groups (and thus the space usage) is upper bounded by $2p_z m$ with probability 0.999. We again assume that $F_0 \ge \sqrt{m}$, since otherwise we can store all groups in $\mathcal{G}$. Under this assumption, we have $p_z \le \frac{11}{\sqrt{m}}$. Therefore, with probability 0.999, the space usage (excluding that for running Algorithm 1) is bounded by $2p_z m = O(\sqrt{m})$. By Theorem 2.1, maintaining a 1.1-approximation of F_0 also costs space $O(\sqrt{m})$. Therefore, the total space is bounded by $O(\sqrt{m})$ with probability 0.99. □

4 General Datasets

Real-world datasets may not be well-shaped. In this section, we develop algorithms for robust F_0-estimation for datasets with small F_0-ambiguity. To this end, we first design an algorithm for robust ℓ_0-sampling on well-shaped datasets where the representative element q_G of every sampled group G does *not* change during the entire streaming process. Note that in all the previous algorithms (Algorithm 1 and 2), the representative elements q_G may change over time, which is due to the fact that when z increases (or, the global sampling probability decreases), some groups may be deleted and then inserted again later. We then show that our new ℓ_0-sampling algorithm can be used for robust F_0-estimation on general datasets.

4.1 An Alternative Algorithm for Robust ℓ_0-Sampling on Well-Shaped Datasets

Our alternative algorithm for ℓ_0-Sampling is described in Algorithm 3. Different from Algorithm 2, it does not use F_0-estimation as a subroutine.

The idea of the new ℓ_0-sampling algorithm is to guess the best sample rate $p^* \approx \frac{1}{F_0}$. To this end, we essentially run Algorithm 2 for $\log m$ times in parallel *without* the F_0-estimation (Line 4 of Algorithm 2); each run uses a fixed sampling threshold $z = \log \frac{1}{p}$, where p takes values $1, \frac{1}{2}, \frac{1}{4} \ldots, \frac{1}{m}$ for the $\log m$ runs. For each run, if $p < p^*$, then we may not be able to obtain a sample. However, in this case, pF_0 is small and we will not waste too much space. Otherwise if $p > p^*$, then pF_0 could be very large. We thus set a space cap $10\sqrt{m}$, and *suspend* the run when the number of stored groups exceeds the cap.

We have the following result.

Theorem 4.1. *Algorithm 3 solves robust ℓ_0-sampling on a well-shaped dataset of m elements using $O(\sqrt{m}\log m)$ words of space; the algorithm succeeds with probability* 0.99.

Proof. Let $M \triangleq 10\sqrt{m}$ be a space parameter. We again assume that $F_0 \ge M$, since otherwise we can store all groups in $\mathcal{G}$.

When $p \ge \min\left\{\frac{10}{F_0}, 1\right\}$, letting $z = \log \frac{1}{p}$ (ignoring ceiling/floor),

$$\left|\mathcal{G}_z^{\text{accept}}\right| \ge 1 \tag{31}$$

with probability at least $1-(1-p)^{F_0} \ge 1-\left(1-\frac{10}{F_0}\right)^{F_0} \ge 0.999$.

Under the assumption that $F_0 \ge M$, we have

$$\mathbf{E}\left[\left|\mathcal{G}_z^{\text{sample}}\right|\right] \le \frac{m}{2^z} \le pm \le \frac{10m}{F_0} \le \frac{10m}{M} = \sqrt{m}.$$

Algorithm 3: Alternative-Robust-ℓ_0-Sampling

Input: a stream of well-shaped dataset Q
Output: an element of a group randomly sampled from $\mathcal{G}(Q)$

$m \leftarrow 0, \zeta \leftarrow 0$
foreach *element q in the stream* **do**
  $m \leftarrow m + 1$
  /* create the ζ-th run when $m > 2^\zeta$ by copying all groups G in $\mathcal{G}^{\text{sample}}_{\zeta-1}$ to $\mathcal{G}^{\text{sample}}_{\zeta}$ if $\texttt{zero}(q^{\max}_G) \geq \zeta$ */
  if $m > 2^\zeta$ **then**
    $\zeta \leftarrow \zeta + 1, \mathcal{G}^{\text{sample}}_{\zeta} \leftarrow \emptyset$
    foreach $G \in \mathcal{G}^{\text{sample}}_{\zeta-1}$ **do**
      if $\texttt{zero}(q^{\max}_G) \geq \zeta$ **then**
        add $(q^{\max}_G, q^{\max}_G, q^{\text{last}}_G)$ to $\mathcal{G}^{\text{sample}}_{\zeta}$
  for $z = 0, 1, \ldots, \zeta$ **do**
    /* each run follows the same procedure as Line 5 - 16 in Algorithm 2, except that we include a point $q^{\max}_G$ in the group description, which is used to determine whether to copy G when initiating a new run; we suspend a run if the number of sampled groups in the run exceeds the limit */
    if $\exists G \in \mathcal{G}^{\text{sample}}_z$ *s.t.* $d(q_G, q) \leq 1$ **then**
      $q^{\text{last}}_G \leftarrow q$
      if $\texttt{zero}(q) > \texttt{zero}(q^{\max}_G)$ **then** $q^{\max}_G \leftarrow q$
    else if $\texttt{zero}(q) \geq z$ **then**
      create a tuple $G \triangleq (q_G, q^{\max}_G, q^{\text{last}}_G) \leftarrow (q, q, q)$ and add G to $\mathcal{G}^{\text{sample}}_z$
      if $\left|\mathcal{G}^{\text{sample}}_z\right| > 10\sqrt{m}$ **then** suspend the z-th run
/* at the time of query: */
for $z = 0, 1, \ldots, \zeta$ **do**
  if *the z-th run is not suspended* **then**
    for $G \in \mathcal{G}^{\text{sample}}_z$ **do**
      if $\texttt{zero}(q^{\text{last}}_G) \geq z$ **then** add q_G to $\mathcal{G}^{\text{accept}}_z$
Let z^* be an arbitrary value z such that the z-th run is not suspended *and* $\left|\mathcal{G}^{\text{accept}}_z\right| \geq 1$
if *no such z^* exists* **then return** "failure"
else return a random element in $\mathcal{G}^{\text{accept}}_{z^*}$

By a Chernoff bound, with probability $1 - e^{-\Omega(\sqrt{m})}$, it holds that

$$\left|\mathcal{G}^{\text{sample}}_z\right| \leq 2\sqrt{m} < M. \tag{32}$$

By a union bound, we have with probability 0.999, $\left|\mathcal{G}^{\text{sample}}_z\right| < M$ holds for all the m time steps of the streaming process.

By (31) and (32), we know that $z = \log \frac{1}{p}$ with $p = \min\left\{\frac{10}{F_0}, 1\right\}$ satisfies both requirements at Line 23 of Algorithm 3, giving the correctness of the algorithm. The total space for all $z = 0, 1, \ldots, \log m$ runs is bounded by $O(\sqrt{m}\log m)$. □

Compared with Algorithm 2, the space cost of Algorithm 3 is higher by a $\log m$ factor. This extra cost is traded for the stability of the representative element q_G of all $G \in \mathcal{G}_z^{\text{sample}}$.

4.2 Robust F_0 for General Datasets

We now slightly modify Algorithm 3 for F_0-estimation on a dataset Q with F_0-ambiguity $\tau \in [0, 0.99]$, as follows:

- At Line 17, replace the budget $10\sqrt{m}$ with $\frac{100}{\epsilon}\sqrt{m}$.
- At the time of query, replace Line 23-25 with: *Find the smallest z^* such that the z^*-th run is not suspended; return $2^{z^*}\left|\mathcal{G}_{z^*}^{\text{accept}}\right|$ as the estimation of F_0.* If no such z^* exists, return "fail".

We have the following theorem.

Theorem 4.2. *Algorithm 3, after the above modifications, computes a $\frac{1+\epsilon}{1-\tau}$-approximation of the robust F_0 of a dataset of m elements with F_0-ambiguity $\tau \in [0, 0.99]$. The algorithm succeeds with probability 0.99 and uses $O\left(\frac{\sqrt{m}}{\epsilon}\log m\right)$ words of space.*

Proof Ideas. The idea behind the proof is to show that once the representative element of each sampled group in our algorithm remains unchanged during the streaming process, there exists a group partition $\mathcal{H}$ of Q such that our algorithm is effectively performing ℓ_0-sampling on the groups in $\mathcal{H}$. Moreover, the cardinality of $\mathcal{H}$ will not deviate from $F_0(Q)$ by more than a τF_0 additive factor. We also show that such a group partition $\mathcal{H}$ can be constructed in a greedy fashion.

Proof. Let $O \subseteq Q$ be a set of outliers such that $\tilde{Q} \triangleq Q \backslash O$ is well-shaped and $F_0(\tilde{Q}) \geq (1-\tau)F_0(Q)$. Let $\tilde{\mathcal{G}}$ be the natural partition of $\tilde{Q}$. By the definition of F_0-ambiguity, we have $\left|\tilde{\mathcal{G}}\right| \geq (1-\tau)F_0$.

Observe that in Algorithm 3, for the z-th run which is not suspended, whenever we sample an element q with $\texttt{zero}(q) \geq z$, we effectively create a group $H \triangleq H(q)$ that contains all elements $q' \in Q$ such that (1) $d(q', q) \leq 1$, and (2) there does not exist a previously formed group $H \in \mathcal{G}^{\text{sample}}$ satisfying $d(q, q_H) \leq 1$. Let $\mathcal{H}$ be the set of groups created in this run. We conceptually extend $\mathcal{H}$ to cover all elements in Q, as follows:

(1) Initialize $Q' \leftarrow Q \backslash \cup_{H\in\mathcal{H}} H$;
(2) Whenever $Q' \neq \emptyset$
  (a) pick an arbitrary element $q \in Q'$, create a new group $H \leftarrow \texttt{Ball}(q, 1)$, where $\texttt{Ball}(q, 1)$ contains all points in Q' within a distance of 1 from q; we call q the representative element of H;
  (b) $\mathcal{H} \leftarrow \mathcal{H} \cup \{H\}$; $Q' \leftarrow Q' \backslash H$.

Since $\tilde{Q}$ is well-shaped, no group in $\mathcal{H}$ can cover elements in two distinct groups in $\tilde{\mathcal{G}}$. Therefore

$$|\mathcal{H}| \geq \left|\tilde{\mathcal{G}}\right| \geq (1-\tau)F_0. \tag{33}$$

On the other hand, we have the following claim, which can be proved by an inductive argument.

Claim 1. $|\mathcal{H}| \leq |\mathcal{G}| = F_0$.

Proof of Claim 1. Let $H_1, \ldots, H_n$ be the groups in $\mathcal{H}$, ordered according to the time of their creation in the streaming process and our conceptual extension. And let $q_1, \ldots, q_n$ be their representative elements. We show that there is an order of groups in $\mathcal{G}$, denoted by $G_1, \ldots, G_{F_0}$, such

that for any $i \in [F_0]$, it holds that

$$\bigcup_{j\in[i]} G_j \subseteq \bigcup_{j\in[i]} H_j. \tag{34}$$

Note that (34) immediately implies $n \le F_0$.

We prove (34) by induction. For the base case, let G_1 be the group in $\mathcal{G}$ that contains q_1. By the definition of robust F_0, we have $G_1 \subseteq \texttt{Ball}(q_1, 1) = H_1$. Now assume that for $k = i-1$, we have

$$\bigcup_{j\in[i-1]} G_j \subseteq \bigcup_{j\in[i-1]} H_j. \tag{35}$$

For $k = i$, we first observe that the representative element q_i of group H_i satisfies $q_i \notin \cup_{j\in[i-1]} H_j$, which, combined with (35), gives $q_i \notin \cup_{j\in[i-1]} G_j$. Let G_i be the group in $\mathcal{G}$ that contains q_i. Clearly,

$$G_i \subseteq \texttt{Ball}(q_i, 1) \subseteq \bigcup_{j\in[i]} H_j. \tag{36}$$

Combining (35) and (36), we have $\cup_{j\in[i]} G_j \subseteq \cup_{j\in[i]} H_j$, which completes the induction. □

By (33) and Claim 1, we have $|\mathcal{H}| \in [(1-\tau)F_0, F_0]$. Note that Algorithm 3 effectively performs ℓ_0-sampling on groups in $\mathcal{H}$. We can thus use Algorithm 3 to estimate $|\mathcal{H}|$. Let $M \triangleq \frac{100}{\epsilon}\sqrt{m}$ be the space parameter. W.l.o.g., assume that $F_0 \ge M$. Let $z_0 \triangleq \log \frac{\epsilon^2|\mathcal{H}|}{40}$ (ignoring ceiling/floor). We note that the algorithm does not need to know z_0, which is only used for the purpose of analysis.

For any z-th run with $z \le z_0$, for each $G \in \mathcal{H}$, let X_G be the indicator variable of the event that $\texttt{zero}(q_G^{\text{last}}) \ge z$, and let $X = \sum_{G\in\mathcal{G}} X_G$. We have $\mathbf{E}[X] = \frac{|\mathcal{H}|}{2^z} \ge \frac{|\mathcal{H}|}{2^{z_0}} = \frac{40}{\epsilon^2}$. By a Chernoff bound, we have $\Pr[|X - \mathbf{E}[X]| \le \epsilon\mathbf{E}[X]] \ge 1 - 2\exp\left(-\frac{\epsilon^2\mathbf{E}[X]}{3}\right) \ge 0.999$. On the other hand, by another Chernoff bound, with probability 0.999, the space usage with respect to the z_0-th run is bounded by $2 \cdot \frac{m}{2^{z_0}} = \frac{80m}{\epsilon^2|\mathcal{H}|} \le \frac{80m}{\epsilon^2(1-\tau)F_0} \le \frac{80m}{\epsilon^2(1-\tau)M} \le M$, where the last inequality uses the fact $\tau \in [0, 0.99]$.

Therefore, for the smallest run z^* such that the z^*-th run is not suspended, the value $2^{z^*}X$ is a $(1+\epsilon)$-approximation of $|\mathcal{H}|$, which is also a $\frac{1+\epsilon}{1-\tau}$-approximation of F_0.

The total space over all $\Theta(\log m)$ runs is bounded by $O(M \log m) = O\left(\frac{\sqrt{m}}{\epsilon}\log m\right)$. □

Remark 1. *The best definition for ℓ_0-sampling on general datasets is unclear, as the minimum-cardinality valid partition may not be unique. Nevertheless, Algorithm 3 still gives the following guarantee: Let Q be a dataset with F_0-ambiguity τ. Let $\tilde{\mathcal{G}} = \{G_1, \ldots, G_n\}$ be the natural partition of $\tilde{Q} = Q \backslash O$, where O is the set of outliers and $n \in [(1-\tau)F_0, F_0]$ according to the definition of F_0-ambiguity. Algorithm 3 outputs an element q such that for each $i \in [n]$, $\Pr[q \in G_i] \in \left[\frac{1}{F_0}, \frac{1}{(1-\tau)F_0}\right]$.*

5 Lower Bound for Robust F_0

In this section, we prove the following lower bound for approximating robust F_0 in the data stream model.

Theorem 5.1. *Any randomized streaming algorithm that computes a 1.1-approximation of the robust F_0 problem on a dataset of m points with probability 0.51 needs at least $\Omega(\sqrt{m})$ bits of space.*

We prove the theorem via a reduction from the Boolean Hidden Matching problem [6, 21, 28]. In Boolean Hidden Matching, denoted by BHM_n where n is a parameter, we have two parties Alice and Bob. Alice holds a column vector $x \in \{0,1\}^n$, and Bob holds a matching M of $\frac{n}{2}$ edges and a column vector $w \in \{0,1\}^{\frac{n}{2}}$. Representing the matching as a matrix M of size $\frac{n}{2} \times n$, where the k-th row corresponds to the k-th edge (i, j) of the matching M as follows: $M_{ki} = M_{kj} = 1$, and $M_{k\ell} = 0$ for any $\ell \in [n]\backslash\{i, j\}$. Under the promise that we either have $Mx = w$ (*yes instance*) or $Mx = \bar{w}$ (*no*

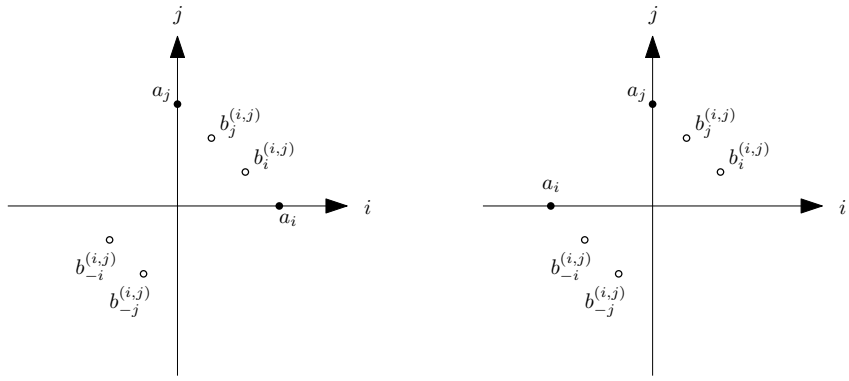


Fig. 1. Hard inputs for robust F_0 in two dimensions w.r.t. an edge $(i, j) \in M$

instance; $\bar{w}$ denotes the vector by flipping each coordinate of $w \in \{0,1\}^{\frac{n}{2}}$), the task is for Alice to send a message to Bob in such a way that Bob can determine whether the input is a yes instance or a no instance. Here, we use modulo 2 arithmetic in the matrix-vector multiplication. We will omit the subscript n in BHM_n when it is clear from the context.

We have the following result for BHM.

Theorem 5.2 ([21]). *There is an input distribution μ for BHM_n, such that for any deterministic one-way communication algorithm that solves the BHM_n problem for at least 0.51 fraction of inputs distributed according to μ, Alice needs to send Bob a message of at least $\Omega(\sqrt{n})$ bits.*

We will use a reduction to show that a streaming algorithm for the robust F_0 problem can be used to create an algorithm for the BHM problem.

Reduction. Given an input $I = (x, M, w)$ of BHM_n, for each $i \in [n]$, let $z_i = 1$ if $x_i = 1$, and $z_i = -1$ if $x_i = 0$; that is, we try to convert $\{0,1\}$ values to $\{-1,1\}$ values. Similarly, for each $i \in [n]$, let $v_i = 1$ if $w_i = 1$, and $v_i = -1$ if $w_i = 0$. Let $\mathbf{e_1}, \ldots, \mathbf{e_n}$ be the standard basis vectors in the n-dimensional Euclidean space. We also identify a vector with a corresponding point in the Euclidean space.

The reduction from BHM to robust F_0 works as follows. Please also refer to Figure 1 for an illustration.

(1) Alice, using her input x, constructs "a" points in $\mathbb{R}^n$ as follows: For each x_i ($i \in [n]$), we create a point $a_i = \frac{3z_i}{\sqrt{2}}\mathbf{e_i}$ in $\mathbb{R}^n$ (recall that z_i is determined by x_i). Alice simulates the streaming algorithm for robust F_0-estimation on the set of n points $\{a_i\}_{i\in[n]}$ in an arbitrary order, and then sends the final memory configuration to Bob.
(2) Bob, using his input M, constructs the following "b" points: For each k-th edge $(i, j) \in M$, create four points in the plane spanned by $\mathbf{e_i}$ and $\mathbf{e_j}$:

$$b_j^{(i,j)} = -\frac{v_k}{\sqrt{2}}\mathbf{e_i} + \frac{2}{\sqrt{2}}\mathbf{e_j}, \quad b_i^{(i,j)} = -\frac{2v_k}{\sqrt{2}}\mathbf{e_i} + \frac{1}{\sqrt{2}}\mathbf{e_j},$$
$$b_{-j}^{(i,j)} = \frac{v_k}{\sqrt{2}}\mathbf{e_i} - \frac{2}{\sqrt{2}}\mathbf{e_j}, \quad b_{-i}^{(i,j)} = \frac{2v_k}{\sqrt{2}}\mathbf{e_i} - \frac{1}{\sqrt{2}}\mathbf{e_j}.$$

(recall that v_i is determined by w_i). He then simulates the streaming algorithm on the set of $2n$ points $\left\{b_i^{(i,j)}, b_j^{(i,j)}, b_{-i}^{(i,j)}, b_{-j}^{(i,j)}\right\}_{(i,j)\in M}$ in an arbitrary order using the memory configuration received from Alice as the starting configuration. Let

$$I' = \{a_i\}_{i\in[m]} \cup \left\{b_i^{(i,j)}, b_j^{(i,j)}, b_{-i}^{(i,j)}, b_{-j}^{(i,j)}\right\}_{(i,j)\in M}.$$

denote the input of $m = 3n$ points to the streaming algorithm. Let $\tilde{F}_0$ be the output of a 1.1-approximation streaming algorithm for computing $F_0(I')$. Bob outputs *Yes* if $\tilde{F}_0 \geq 1.7n$ and *No* if $\tilde{F}_0 < 1.7n$ for BHM(I).

Intuitively, in the yes instance, two of the four "b" points created by Bob lie in the quadrant defined by a_i, the origin, and a_j, while the other two are simply their reflections across the origin. The yes instance in Fig. 1 illustrates just one of the four possible scenarios for the yes instance. On the other hand, in the no instance, none of the four "b" points created by Bob fall within the quadrant a_i-origin-a_j. The no instance figure in Fig. 1 also shows just one of the four possible scenarios for the no instance.

Lemma 5.3. *If there is a streaming algorithm that givens a* 1.1*-approximation of the* $F_0(I')$ *using* S *bits of space, then the corresponding one-way communication algorithm described above solves* BHM(I) *using* S *bits of communication .*

Proof. In the *yes* instance (cf. Figure 1), the minimum-cardinality F_0-partition is as follows:

(1) For each $i \in [n]$, form a group $G_i = \{a_i\} \cup \{\text{all "}b\text{" points in } \texttt{Ball}(a_i, 1)\}$.
(2) After the first step, for each $(i, j) \in M$, form a group that contains two remaining "b" points, whose distance is 1.

The total number of groups is $1.5n$.

In the *no* instance, one of the minimum-cardinality F_0-partitions is as follows:

(1) For each $i \in [n]$, form a group $G_i = \{a_i\} \cup \{\text{all "}b\text{" points in } \texttt{Ball}(a_i, 1)\}$.
(2) After the first step, for each $(i, j) \in M$, form a group for each of the two remaining "b" points. Note that the distance of these two points is bigger than 1.

The total number of groups is $2n$.

Now, if $\tilde{F}_0$ is an approximation of F_0 up to a factor of 1.1, then in the yes instance, we always have $\tilde{F}_0 \leq 1.5n \cdot 1.1 = 1.65n$. While in the no instance, we always have $\tilde{F}_0 \geq 2n \cdot 0.9 = 1.8n$. Since there is a gap between the two "boundary" values $1.65n$ and $1.8n$, we can set the threshold to $1.7n$ and check whether $\tilde{F}_0 > 1.7n$ or $\tilde{F}_0 < 1.7n$ to determine if we have a yes BHM instance or a no instance. □

Theorem 5.1 follows from Theorem 5.2, Lemma 5.3, and Yao's minimax principle [35].

6 Constant Dimensional Euclidean Spaces

In this section, we consider $O(1)$-dimensional Euclidean spaces. We show that our results for F_0-estimation and ℓ_0-sampling for general metric spaces can be further improved to match the results in [9, 10].

6.1 Robust F_0-Estimation

We try to modify Algorithm 1 to get a streaming algorithm for robust F_0 of a well-shaped dataset in the $O(1)$-dimensional Euclidean space. We first partition the Euclidean space using a square grid $\mathcal{C}$ of side length $1/\sqrt{d}$. Thus for each cell $C \in \mathcal{C}$, there exists at most one group $G \in \mathcal{G}$ such that $C \cap G \neq \emptyset$. On the other hand, for each group $G \in \mathcal{G}$, at most $\kappa \triangleq \kappa(d) = O(1)$ grid cells C satisfy

Algorithm 4: ROBUST-F_0-EUCLIDEAN

Input: a stream of well-shaped dataset Q in d-dimensional Euclidean space with $d = O(1)$, parameter ϵ

Output: a $(1+\epsilon)$-approximation of $F_0(Q)$

$\varepsilon \leftarrow \epsilon/(4\kappa)$, $z \leftarrow 0$, $\mathcal{G}^{\text{sample}} \leftarrow \emptyset$

foreach *point q in the stream* **do**

  /* if q belongs to some group G, try to maintain the representative cell C_{q_G} as a random cell from those cells that G intersect with the highest zero$(\cdot)$ value */

  if $\exists G \in \mathcal{G}^{\text{sample}}$ *s.t.* $d(q_G, q) \le 1$ **then**

    if $\nexists C \in C_G$ *s.t.* $q \in C$ **then**

      add C_q to C_G

      if $\texttt{zero}(C_q) > \texttt{zero}(C_{q_G})$ **then**

        $q_G \leftarrow q$, $C_G \leftarrow \{C_q\}$, $r_G \leftarrow 1$

      else if $\texttt{zero}(C_q) = \texttt{zero}(C_{q_G})$ **then**

        $r_G \leftarrow r_G + 1$

        w.pr. $\frac{1}{r_G}$, $q_G \leftarrow q$, $C_G \leftarrow \{C_q\}$ /* Reservoir sampling */

  else if $\texttt{zero}(C_q) \ge z$ **then**

    /* for a newly sampled element that doesn't belong to any existing group, create a new group $G = (q, \{C_q\}, 1)$ and add it to $\mathcal{G}^{\text{sample}}$ */

    create a tuple $G \triangleq (q_G, C_G, r_G) \leftarrow (q, \{C_q\}, 1)$ and add G to $\mathcal{G}^{\text{sample}}$

    if $\sum_{G \in \mathcal{G}^{\text{sample}}} |C_G| > \frac{10^3 \kappa \log \kappa}{\epsilon^2}$ **then**

      $z \leftarrow z + 1$

      /* if representative cell C_{q_G} is no longer sampled, delete G */

      foreach $G \in \mathcal{G}^{\text{sample}}$ **do**

        if $\texttt{zero}(C_{q_G}) < z$ **then** delete G from $\mathcal{G}^{\text{sample}}$

/* at the time of query */

for $j = 1, \ldots, \kappa$ **do**

  Let X_j be the number of groups $G \in \mathcal{G}^{\text{sample}}$ with $|C_G| = j$

return $\sum_{j \in [\kappa-1]} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_\kappa}{\lambda_\kappa}$

$C \cap G \neq \emptyset$. After this setup, we can effectively *ignore* all points $q \in G$ that are not the first point that falls into the cell C_q, where C_q is the cell that contains point q.

Similar as before, we define a hash function $h : \mathcal{C} \to [N]$, where $N = 2^\lambda$ for a large enough integer λ. We again assume that there is no collision among the hash values $\{h(C) \mid C \in \mathcal{C}\}$, which holds with probability $1 - o(1)$ when $N \ge m^3$ (or, $\lambda \ge 3 \log m$), where m is the number of elements in the data stream. Let $\texttt{zero}(C)$ $(C \in \mathcal{C})$ be the number of trailing zeros of $h(C)$.

The Algorithm. Our algorithm is described in Algorithm 4. During the whole streaming process, we maintain a set of sampled groups G in the format (q_G, C_G, r_G), where q_G is a random point of G that has the largest $\texttt{zero}(\cdot)$ value, C_G is the set of cells that contain at least one point in G that comes after q_G, and r_G is the counter that we maintain for performing Reservoir sampling. The

main differences between Algorithm 4 and Algorithm 1 are (1) we consider $\texttt{zero}(\cdot)$ of cells instead of points, and (2) in Algorithm 4, we explicitly maintain all cells that contain at least one point in G that comes after q_G instead of just the number of such cells.

For each incoming point q in the data stream, if it belongs to a sampled group $G \in \mathcal{G}^{\text{sample}}$, we first check if it is the first point in the cell C_q. If yes, we add C_q to C_G. We also try to update q_G if $\texttt{zero}(C_q) \geq \texttt{zero}(C_{q_G})$, in the same way as that in Algorithm 1.

If q does not belong to any group in $\mathcal{G}^{\text{sample}}$ and $\texttt{zero}(C_q) \geq z$, we create a new sampled group. If the sum of the numbers of cells in C_G across all sampled groups G exceeds $\frac{10^3\kappa\log\kappa}{\varepsilon^2}$, we increment the value z and delete all groups $G \in \mathcal{G}^{\text{sample}}$ for which $\texttt{zero}(C_{q_G}) < z$.

We have the following theorem.

Theorem 6.1. *Algorithm 4 outputs a $(1+\epsilon)$-approximation of the robust F_0 of a well-shaped dataset of m points in the $O(1)$-dimensional Euclidean space with probability 0.99 using $O\left(\frac{1}{\epsilon^2}\right)$ words of space.*

In the rest of this section, we prove Theorem 6.1. Let $W \triangleq \frac{10^3\kappa\log\kappa}{\varepsilon^2}$, and let $M = 2\kappa W = O\left(\frac{1}{\varepsilon^2}\right)$ be the space budget (recall that $\kappa = O(1)$). W.l.o.g., we assume that $F_0 \geq M$, since otherwise we can store all groups in the memory. Let $p_z \triangleq \frac{1}{2^z}$ be the sample rate at the end of the streaming process.

We have the following lemma, which we condition on in the rest of the proof.

Lemma 6.2. *With probability 0.998, it holds that $p_z F_0 \in \left[\frac{W}{6\kappa}, 2W\right]$.*

Proof. For the lower bound, if on the contrary that $p_z F_0 < \frac{W}{6\kappa}$, then $p_{z-1}F_0 < \frac{W}{3\kappa}$. Thus, before the last increment of z, we have

$$\mathbf{E}\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G|\right] \leq \kappa \cdot p_{z-1}F_0 \leq \frac{W}{3} = \frac{10^3\kappa\log\kappa}{3\varepsilon^2}.$$

By Lemma 1.6, we have

$$\Pr\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G| > \frac{10^3\kappa\log\kappa}{\varepsilon^2}\right] \leq \exp\left(-\left(\frac{10^3\kappa\log\kappa}{\varepsilon^2} - \frac{2\cdot 10^3\kappa\log\kappa}{3\varepsilon^2}\right)\right) \leq 0.001.$$

In other words, with probability 0.999, the last increment would not happen. A contradiction.

For the upper bound, if on the contrary that $p_z F_0 > 2W$, then

$$\mathbf{E}\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G|\right] \geq p_z F_0 > 2W = \frac{2\cdot 10^3\kappa\log\kappa}{\varepsilon^2}.$$

By a Chernoff bound, we have

$$\begin{aligned}\Pr\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G| \leq \frac{10^3\kappa\log\kappa}{\varepsilon^2}\right] &\leq \Pr\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G| \leq \frac{1}{2}\cdot\mathbf{E}\left[\sum_{G\in\mathcal{G}^{\text{sample}}} |C_G|\right]\right] \\ &\leq \exp\left(-\frac{1}{12}\cdot\frac{2\cdot 10^3\kappa\log\kappa}{\varepsilon^2}\right) \\ &\leq 0.001.\end{aligned}$$

Therefore, with probability 0.999, z would have already been incremented. A contradiction. □

Similar to the analysis for Algorithm 1, we define the following quantities for any $j \in [\kappa]$.

- n_j: the number of groups $G \in \mathcal{G}$ such that G intersects with j cells.
- X_j: the number of groups $G \in \mathcal{G}^{\text{sample}}$ with $|C_G| = j$.
- λ_j: $\lambda_j \triangleq \frac{1-(1-p_z)^j}{j}$; we have
$$\lambda_1 > \lambda_2 > \ldots > \lambda_\kappa, \tag{37}$$
and $\forall i \in [\kappa]$,
$$\lambda_i \in [\lambda_\kappa, \lambda_1] \subseteq \left[p_z - \frac{\kappa}{2}p_z^2, p_z\right] \subseteq \left[\frac{p_z}{2}, p_z\right], \tag{38}$$
where the last relation, by $p_z F_0 \le 2W$ (Lemma 6.2), $M = 2\kappa W$, and the assumption that $F_0 \ge M$, we have $\kappa p_z \le 1$.

By essentially the same calculation as (8), we get
$$\mathbf{E}[X_j] = \lambda_j n_j + \ldots + \lambda_\kappa n_\kappa. \tag{39}$$

We thus define our estimator for $|\mathcal{G}|$ as
$$\tilde{F}_0 \triangleq \sum_{j\in[\kappa-1]} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_\kappa}{\lambda_\kappa}.$$

It is easy to see that
$$\mathbf{E}[\tilde{F}_0] \triangleq \sum_{j\in[\kappa-1]} \frac{\mathbf{E}[X_j] - \mathbf{E}[X_{j+1}]}{\lambda_j} + \frac{\mathbf{E}[X_\kappa]}{\lambda_\kappa} = \sum_{j\in[\kappa]} n_j = F_0. \tag{40}$$

In the rest of the proof, we show that $\tilde{F}_0$ is tightly concentrated on its expectation $\mathbf{E}[\tilde{F}_0]$.

Similar as before, let j^* be the index such that $\mathbf{E}[X_{j^*}] > \varepsilon\lambda_\kappa F_0$ and $\mathbf{E}[X_{j^*+1}] \le \varepsilon\lambda_\kappa F_0$. We write the output of Algorithm 4 as $U + V$, where
$$U = \sum_{j\in[j^*-1]} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_{j^*}}{\lambda_{j^*}}, \tag{41}$$
and
$$V = -\frac{X_{j^*+1}}{\lambda_{j^*}} + \sum_{j=j^*+1,\ldots,\kappa-1} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_\kappa}{\lambda_\kappa}.$$

We first analyze U:
$$\mathbf{E}[U] = \sum_{j\in[j^*-1]} \frac{\mathbf{E}[X_j] - \mathbf{E}[X_{j+1}]}{\lambda_j} + \frac{\mathbf{E}[X_{j^*}]}{\lambda_{j^*}} = \sum_{j\in[j^*-1]} \left(\frac{1}{\lambda_{j+1}} - \frac{1}{\lambda_j}\right)\mathbf{E}[X_{j+1}] + \frac{\mathbf{E}[X_1]}{\lambda_1}. \tag{42}$$

Set $\eta = \varepsilon\lambda_\kappa F_0$. By the definition of j^*, we know that $\mathbf{E}[X_j] > \varepsilon\lambda_\kappa F_0$ for any $j \in [j^*]$. Therefore, for any $j \in [j^*]$, $\frac{\eta}{\mathbf{E}[X_j]} < 1$. On the other hand, by (39) we have $\mathbf{E}[X_j] \le \lambda_1 F_0$ for any $j \in [\kappa]$. Therefore, $\frac{\eta}{\mathbf{E}[X_j]} \ge \frac{\varepsilon\lambda_\kappa F_0}{\lambda_1 F_0} = \frac{\varepsilon\lambda_\kappa}{\lambda_1}$. By a Chernoff bound, we have for any $j \in [j^*]$,
$$\begin{aligned}
\Pr\left[\left|X_j - \mathbf{E}[X_j]\right| \ge \eta\right] &\le 2\exp\left(-\frac{\eta^2}{3\mathbf{E}[X_j]}\right) \le 2\exp\left(-\frac{\varepsilon^2\lambda_\kappa^2 F_0}{3\lambda_1}\right) \\
&\overset{(38)}{\le} 2\exp\left(-\frac{\varepsilon^2 p_z F_0}{12}\right) \\
&\overset{\text{Lemma 6.2}}{\le} 2\exp\left(-\frac{\varepsilon^2 W}{12\cdot 6\kappa}\right) = 2\exp\left(-\frac{10^3\kappa\log\kappa}{12\cdot 6\kappa}\right) \\
&\le \frac{0.001}{\kappa}.
\end{aligned} \tag{43}$$

By a union bound over all $j \in [j^*]$, we have

$$\begin{aligned}
\Pr\left[|U - \mathbf{E}[U]| \le \frac{2\kappa\eta}{p_z}\right] &\overset{(38)}{\ge} \Pr\left[|U - \mathbf{E}[U]| \le \frac{\kappa\eta}{\lambda_k}\right] \\
&\overset{(37),\,(41),\,(42)}{\ge} 1 - \sum_{j\in[j^*]} \Pr\left[\left|X_j - \mathbf{E}[X_j]\right| \ge \eta\right] \\
&\overset{(43)}{\ge} 1 - \kappa \cdot \frac{0.001}{\kappa} \\
&= 0.999. \qquad (44)
\end{aligned}$$

We next bound V. We have

$$0 \le V \le \sum_{j=j^*+1,\ldots,\kappa-1} \frac{X_j - X_{j+1}}{\lambda_j} + \frac{X_\kappa}{\lambda_\kappa} \le \frac{1}{\lambda_\kappa} \cdot X_{j^*+1}.$$

By the definition of j^*, we have $\mathbf{E}[X_{j^*+1}] \le \varepsilon\lambda_\kappa F_0$. By Lemma 6.2, it holds that

$$\begin{aligned}
\Pr\left[X_{j^*+1} > 3\varepsilon\lambda_\kappa F_0\right] &\le \exp(-(3\varepsilon\lambda_\kappa F_0 - 2\varepsilon\lambda_\kappa F_0)) \\
&\overset{(38)}{\le} \exp\left(-\frac{\varepsilon p_z F_0}{2}\right) \\
&\overset{\text{Lemma 6.2}}{\le} \exp\left(-\frac{\epsilon W}{12\kappa}\right) \le 0.001.
\end{aligned}$$

Therefore, with probability 0.999,

$$V \le 3\varepsilon F_0. \qquad (45)$$

Set $\varepsilon = \epsilon/(4\kappa)$. By (44) and (45), we have that with probability 0.99, the quantity $\tilde{F}_0$ approximates F_0 up to an additive error $2\kappa\varepsilon F_0 + 3\varepsilon F_0 \le \epsilon F_0$. Since κ is a constant, the space cost is bounded by $M = O\left(\frac{1}{\varepsilon^2}\right) = O\left(\frac{1}{\epsilon^2}\right)$.

6.2 Robust ℓ_0-Sampling

We can slightly modify Algorithm 2 to obtain a streaming algorithm for robust ℓ_0-sampling in the $O(1)$-dimensional Euclidean space with the help of the grid partition as described in Section 6.1. The algorithm is described in Algorithm 5. Compared with Algorithm 2, the difference is that we only keep the first point of each grid cell that intersects with a sampled group.

We have the following theorem.

Theorem 6.3. *Algorithm 5 solves robust ℓ_0-sampling on a well-shaped dataset of m points with probability* 0.99 *using $O(1)$ words of space.*

The proof of Theorem 6.3 is similar to that of Theorem 3.1. We again set $p_z = \frac{10}{\tilde{F}_0}$, where $\tilde{F}_0$ is a 1.1-approximation of F_0. This is sufficient to guarantee that $\Pr\left[\left|\mathcal{G}^{\text{accept}}\right| \ge 1\right] \ge 0.999$. On the other hand, the space cost is upper bounded by $2p_z\kappa F_0 = O(1)$ with probability 0.999 by a Chernoff bound, where $\kappa \triangleq \kappa(d) = O(1)$ is the maximum number of grid cells that a group can intersect. On the other hand, maintaining a 1.1-approximation of robust F_0 using Robust-F_0-Euclidean (Algorithm 4) costs space $O(1)$ words. Therefore, the total space cost is $O(1)$ words.

7 Other Statistical Problems

This paper mainly focuses on two basic statistical problems, namely, distinct elements and ℓ_0-sampling. A few other statistical problems can be solved on well-shaped datasets by adapting

Algorithm 5: ROBUST-ℓ_0-SAMPLING-EUCLIDEAN

Input: a stream of well-separated dataset Q in d-dimensional Euclidean space with $d = O(1)$
Output: an element of a group randomly sampled from $\mathcal{G}(Q)$

$z \leftarrow 0$, $\mathcal{G}^{\text{sample}} \leftarrow \emptyset$, $\mathcal{G}^{\text{accept}} \leftarrow \emptyset$
foreach *point q in the stream* **do**
  $\tilde{F}_0 \leftarrow$ ROBUST-F_0-EUCLIDEAN$(q, 0.1)$, /* $\tilde{F}_0$ is a 1.1-approximation of F_0 */
  $z \leftarrow \lfloor \log \frac{\tilde{F}_0}{10} \rfloor$
  foreach $G \in \mathcal{G}^{\text{sample}}$ **do**
    /* if representative cell C_{q_G} is no longer sampled, delete G */
    if $\texttt{zero}(C_{q_G}) < z$ **then** delete G from $\mathcal{G}^{\text{sample}}$
  if $\exists G \in \mathcal{G}^{\text{sample}}$ *s.t.* $d(q_G, q) \le 1$ **then**
    /* if q belongs to some group G, update q_G^{last}, and update q_G if necessary */
    if $\nexists C \in C_G$ *s.t.* $q \in C$ **then**
      add C_q to C_G
      $q_G^{\text{last}} \leftarrow q$
      if $\texttt{zero}(C_q) > \texttt{zero}(C_{q_G})$ **then** $q_G \leftarrow q$
  else
    /* for a newly sampled element that doesn't belong to any existing group, create a new group $G = (q, q, C_q)$ and add it to $\mathcal{G}^{\text{sample}}$ */
    if $\texttt{zero}(C_q) \ge z$ **then**
      create a tuple $G \triangleq (q_G, q_G^{\text{last}}, C_G) \leftarrow (q, q, C_q)$ and add G to $\mathcal{G}^{\text{sample}}$
/* at the time of query: */
for $G \in \mathcal{G}^{\text{sample}}$ **do**
  if $\texttt{zero}(C_{q_G^{\text{last}}}) \ge z$ **then** add G to $\mathcal{G}^{\text{accept}}$
if $\left|\mathcal{G}^{\text{accept}}\right| = 0$ **then**
  return "failure"
else return a random group G in $\mathcal{G}^{\text{accept}}$

the algorithms for the noiseless setting. For example, consider the following two basic statistical problems:

(1) (α, ϵ)-*heavy-hitters:* identify a set of groups $\mathcal{G}' \subseteq \mathcal{G}$ that contains all groups G with sizes $|G| \ge \alpha m$, but excludes all groups with $|G| \le (\alpha - \epsilon)m$, where m is the length of the stream.
(2) *Frequency moments* F_p $(p \ge 1)$*:* compute $F_p = \sum_{G \in \mathcal{G}} |G|^p$.

The Misra-Gries sketch [31] and the AMS-sketch [4] are two algorithms that solve (α, ϵ)-heavy-hitters and frequency moments F_p $(p \ge 1)$, respectively. A nice feature in both sketching algorithms, translating it to the well-shaped data setting, is that we always maintain a representative element of a subset of groups in the sketch. Therefore, we can easily determine if a newly incoming element e belongs to any of the existing groups G. Consequently, we can adapt the Misra-Gries sketch and the AMS-sketch for the two problems on well-shaped datasets using the same amount of space as on noiseless datasets.

On the other hand, it is not clear how to define some of these statistical problems for general noisy datasets. One approach is to introduce an ambiguity parameter similar to that used in F_0-estimation. We leave this to future work.

8 Concluding Remarks

We conclude the paper by discussing a couple of practical aspects and open problems.

We would like to mention two methods that may further reduce time and space costs of our algorithms in practice. The first is to use the *space partition* technique, first introduced in [9], to further improve the space cost. By partitioning the space into cells in such a way that each cell intersects with at most one group in the F_0-partition of the dataset, we can effectively decrease the number of input elements m to the number of non-empty cells. More precisely, for each incoming element q in the data stream, we feed the cell C_q into our algorithms instead of q, where C_q is the cell that contains q. In Section 6, we have already used this idea to improve the space cost of F_0-estimation and ℓ_0-sampling for points in the $O(1)$-dimensional Euclidean space.

In this work, we are mainly interested in minimizing the space cost of streaming algorithms, and do not attempt to optimize their running time. The time bottleneck in all of our algorithms is the step of membership search. That is, for each incoming element q in the data stream, we need to check whether there is a group $G \in \mathcal{G}^{\text{sample}}$ such that $d(q_G, q) \leq 1$. In metric spaces where efficient LSH schemes are applicable, we can utilize LSH to accelerate the search process, thereby expediting the whole algorithm.

There are several directions left open after this work: (1) As mentioned in Section 5, for robust F_0-estimation, it would be desirable if we can prove a $\Omega(\sqrt{m})$ lower bound for well-shaped datasets (the *no-instance* in the current hard input distribution is not well-shaped) and/or incorporate the approximation factor ϵ into the lower bound. (2) It would be nice if we can design sublinear space algorithms for turnstile data streams where element deletions are allowed. (3) It would be interesting to study other statistical problems for which algorithms for noiseless datasets cannot be directly adapted.

References

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012.

[2] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14, 2012.

[3] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *APPROX-RANDOM*, pages 1–10, 2013.

[4] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[5] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *SODA*, pages 1345–1364, 2016.

[6] Ziv Bar-Yossef, T. S. Jayram, and Iordanis Kerenidis. Exponential separation of quantum and classical one-way communication complexity. In László Babai, editor, *STOC*, pages 128–137. ACM, 2004.

[7] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM*, pages 1–10, 2002.

[8] Kevin Beyer, Peter J Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD*, pages 199–210, 2007.

[9] Di Chen and Qin Zhang. Streaming algorithms for robust distinct elements. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *SIGMOD*, pages 1433–1447. ACM, 2016.

[10] Jiecao Chen and Qin Zhang. Distinct sampling on streaming data with near-duplicates. In Jan Van den Bussche and Marcelo Arenas, editors, *PODS*, pages 369–382. ACM, 2018.

[11] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related

problems in dynamic graph streams. In *SODA*, pages 1326–1344, 2016.
[12] Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *SODA*, pages 1234–1251, 2015.
[13] Graham Cormode, S. Muthukrishnan, and Irina Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *VLDB*, pages 25–36, 2005.
[14] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.
[15] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Algorithms-ESA 2003*, pages 605–617. Springer, 2003.
[16] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
[17] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings*, (1), 2008.
[18] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.
[19] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geometry Appl.*, 18(1/2):3–28, 2008.
[20] Sumit Ganguly. Counting distinct items over update streams. *Theoretical Computer Science*, 378(3):211–222, 2007.
[21] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 516–525. ACM, 2007.
[22] Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *SPAA*, pages 281–291, 2001.
[23] Jeongwan Haah, Aram W. Harrow, Zheng-Feng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal tomography of quantum states. In Daniel Wichs and Yishay Mansour, editors, *STOC*, pages 913–925. ACM, 2016.
[24] Thomas N Herzog, Fritz J Scheuren, and William E Winkler. *Data quality and record linkage techniques*, volume 1. Springer, 2007.
[25] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Jeffrey Scott Vitter, editor, *STOC*, pages 604–613. ACM, 1998.
[26] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *PODS*, pages 49–58, 2011.
[27] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52, 2010.
[28] Iordanis Kerenidis and Ran Raz. The one-way communication complexity of the boolean hidden matching problem. *CoRR*, abs/quant-ph/0607173, 2006.
[29] Christian Konrad. Maximum matching in turnstile streams. In *ESA*, pages 840–852, 2015.
[30] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, pages 802–803. ACM, 2006.
[31] J. Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982.
[32] Noam Nisan. Psuedorandom generators for space-bounded computation. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 204–212. ACM, 1990.
[33] Ryan O'Donnell and John Wright. Efficient quantum tomography. In Daniel Wichs and Yishay Mansour, editors, *STOC*, pages 899–912. ACM, 2016.
[34] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.
[35] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977.
[36] Qin Zhang. Communication-efficient computation on distributed noisy datasets. In *SPAA*, pages 313–322, 2015.

Received December 2024; revised February 2025; accepted March 2025; revised 20 February 2007; revised 12 March 2009; accepted 5 June 2009