Bounded Space Differentially Private Quantiles

Anonymous authors
Paper under double-blind review

Abstract

Estimating the quantiles of a large dataset is a fundamental problem in both the streaming algorithms literature and the differential privacy literature. However, all existing private mechanisms for distribution-independent quantile computation require space at least linear in the input size n. In this work, we devise a differentially private algorithm for the quantile estimation problem, with strongly sublinear space complexity, in the one-shot and continual observation settings. Our basic mechanism estimates any α-approximate quantile of a length-n stream over a data universe $\mathcal{X}$ with probability $1-\beta$ using $O\left(\frac{\log(|\mathcal{X}|/\beta)\log(\alpha\epsilon n)}{\alpha\epsilon}\right)$ space while satisfying ϵ-differential privacy at a single time point. Our approach builds upon deterministic streaming algorithms for non-private quantile estimation instantiating the exponential mechanism using a utility function defined on sketch items, while (privately) sampling from intervals defined by the sketch. We also present another algorithm based on histograms that is especially well-suited to the multiple quantiles case. We implement our algorithms and experimentally evaluate them on synthetic and real-world datasets.

1 Introduction

Quantile estimation is a fundamental subroutine in data analysis and statistics. For $q \in [0,1]$, the q-quantile of a dataset of size n is the element ranked $\lceil qn \rceil$ when the elements are sorted from smallest to largest. Computing a small number of quantiles in a massive data set can serve as a quick and effective sketch of the "shape" of the data. Quantile estimation also serves an essential role in robust statistics, where data is generated from some distribution but is contaminated by a non-negligible fraction of outliers, i.e., "out of distribution" elements that may sometimes even be adversarial. For example, the median (50th percentile) of a dataset is used as a robust estimator of the mean in such situations where the data may be contaminated. Location parameters can also be (robustly) estimated via truncation or winsorization, an operation that relies on quantile estimation as a subroutine (Tukey, 1960; Huber, 1964). Rank-based nonparametric statistics can be used for hypothesis testing (e.g., the Kruskal-Wallis test statistic (Kruskal & Wallis, 1952)). Thus, designing quantile-based or rank-based estimators, whether distribution-dependent or distribution-agnostic, is important in many scenarios.

Maintaining the privacy of individual users or data items, or even of groups, is an essential prerequisite in many modern data analysis and management systems. Differential privacy (DP) is a rigorous and now well-accepted definition of privacy for data analysis and machine learning. In particular, there is already a substantial amount of literature on differentially private quantile estimation (e.g., see Nissim et al. (2007); Asi & Duchi (2020); Gillenwater et al. (2021); Tzamos et al. (2020)).[1]

All previous work either makes certain distributional assumptions about the input, or assume the ability to access all input elements (thus virtually requiring a linear or worse space complexity). Such assumptions may be infeasible in many practical scenarios, where large scale databases have to quickly process streams of millions or billions of data elements without clear a priori distributional characteristics. The field of streaming algorithms aims to provide space-efficient algorithms for data analysis tasks such as these. These algorithms typically maintain good accuracy and fast running time while having space requirements that

[1] Robust estimators are also known to be useful for accurate differentially private estimation; see, e.g., the work of Dwork and Lei (Dwork & Lei, 2009) in the context of quantile estimation for the interquartile range and for medians.

are substantially smaller than the size of the data. While distribution-agnostic quantile estimation is among the most fundamental problems in the streaming literature (Agarwal et al., 2013; Felber & Ostrovsky, 2017; Greenwald & Khanna, 2001; Hung & Ting, 2010; Karnin et al., 2016; Manku et al., 1999; Munro & Paterson, 1980; Shrivastava et al., 2004; Wang et al., 2013), no differentially private sublinear-space algorithms for the same task are currently known. Thus, the following question, essentially posed by (Smith, 2011) and (Mir et al., 2011), naturally arises:

> *Can we design differentially private quantile estimators that use space sublinear in the stream length, have efficient running time, provide high-enough utility, and do not rely on restrictive distributional assumptions?*

It is well-known (Munro & Paterson, 1980) that exact computation of quantiles cannot be done with sublinear space, even where there are no privacy considerations. Thus, one must resort to approximation. Specifically, for a dataset of n elements, an *α-approximate q-quantile* is any element which has rank $(q \pm \alpha)n$ when sorting the elements from smallest to largest, and it is known that the space complexity of α-approximating quantiles is $\tilde{\Omega}(1/\alpha)$ (Munro & Paterson, 1980). In our case, the general goal is to efficiently compute α-approximate quantiles in a (pure or approximate) differentially private manner.

1.1 Our Contributions

We answer the above question affirmatively by providing theoretically proven algorithms with accompanying experimental validation for quantile estimation with DP guarantees. The algorithms are suitable for private computation of either a single quantile or multiple quantiles. Concretely, the main contributions are:

1. We devise `DPExpGK`, a differentially private sublinear-space algorithm for quantile estimation based on the exponential mechanism. In order to achieve sublinear space complexity, our algorithm carefully instantiates the exponential mechanism with the basic blocks being intervals from the Greenwald-Khanna (Greenwald & Khanna, 2001) data structure for non-private quantile estimation, rather than single elements. We prove general distribution-agnostic utility bounds on our algorithm and show that the space complexity is logarithmic in n.
2. We present `DPHistGK`, another differentially private mechanism for quantile estimation, which applies the Laplace mechanism to a histogram, again using intervals of the GK-sketch as the basic building block. We theoretically demonstrate that `DPHistGK` may be useful in cases where one has prior knowledge on the input.
3. We extend our results to the *continual release* setting, wherein we must maintain and output an estimate of the queried quantile in an online manner as the data is received and processed.
4. We empirically validate our results by evaluating `DPExpGK`, analyzing and comparing various aspects of performance on real-world and synthetic datasets.

2 Related Work

2.1 Quantile Approximation of Streams and Sketches

Approximation of quantiles in large data streams (without privacy guarantees) is among the most well-investigated problems in the streaming literature (Wang et al., 2013; Greenwald & Khanna, 2016; Xiang et al., 2020). A classical result of Munro and Paterson from 1980 (Munro & Paterson, 1980) shows that computing the median exactly with p passes over a stream requires $\Omega(n^{1/p})$ space, thus implying the need for approximation to obtain very efficient (that is, at most polylogarithmic) space complexity. Manku, Rajagopalan and Lindsay (Manku et al., 1999) built on ideas from (Munro & Paterson, 1980) to obtain a randomized algorithm with only $O((1/\alpha)\log^2(n\alpha))$ for α-approximating all quantiles; a deterministic variant of their approach with the same space complexity exists as well (Agarwal et al., 2013). The best known deterministic algorithm is that of Greenwald and Khanna (GK) (Greenwald & Khanna, 2001) on which we

build on in this paper, with a space complexity of $O(\alpha^{-1}\log(\alpha n))$ to sketch all quantiles for n elements (up to rank approximation error of $\pm\alpha n$). A recent deterministic lower bound of Cormode and Veselý (Cormode & Veselý, 2020) (improving on (Hung & Ting, 2010)) shows that the GK algorithm is in fact optimal among deterministic (comparison-based) sketches.

Randomization and sampling help for streaming quantiles, and the space complexity becomes independent of n; an optimal space complexity of $O((1/\alpha)\log\log(1/\beta))$ was achieved by Karnin, Lang and Liberty (Karnin et al., 2016) (for failure probability β), concluding a series of work on randomized algorithms (Agarwal et al., 2013; Felber & Ostrovsky, 2017; Luo et al., 2016; Manku et al., 1999).

The problem of biased or relative error quantiles, where one is interested in increased approximation accuracy for very small or very large quantiles, has also been investigated (Cormode et al., 2020; 2006); it would be interesting to devise efficient differentially private algorithms for this problem.

Recall that our approach is based on the Greenwald-Khanna deterministic all-quantiles sketch (Greenwald & Khanna, 2001). While some of the aforementioned randomized algorithms have a slightly better space complexity, differential privacy mechanisms are inherently randomized by themselves, and the analysis seems somewhat simpler and more intuitive when combined with a deterministic sketch. This, of course, does not rule out improved private algorithms based on modern efficient randomized sketches (see Section 7).

2.2 Differential Privacy

Differentially private single quantile estimation: Works by Nissim, Raskhodnikova and Smith (Nissim et al., 2007) and Asi and Duchi (Asi & Duchi, 2020) improve the trade-off between accuracy and privacy by scaling the noise added for obfuscation in an instance-specific manner for median estimation in the absence of distributional assumptions. Another work by Dwork and Lei (Dwork & Lei, 2009) uses a "propose-test-release" paradigm to take advantage of local sensitivity; however, as observed in Gillenwater et al. (2021), in practice the error incurred by this method is relatively large as compared to other works like Tzamos et al. (2020). The work Tzamos et al. (2020) achieves the optimal trade-off between privacy and utility in the distributional setting, but again as observed by Gillenwater et al. (2021), with a time complexity of $O(n^4)$, this method does not scale well to large data sets.

The work Gillenwater et al. (2021) shows how to optimize the division of the privacy budget to estimate m quantiles in a time-efficient manner. For estimation of m quantiles, their time and space complexity are $O(mn\log(n) + m^2 n)$ and $O(m^2 n)$, respectively. They do an extensive experimental analysis and find lower error compared to previous work. However, although they provide intuition for why their method should incur relatively low error, they do not achieve formal theoretical accuracy guarantees. Kaplan et al. (2021) improve upon Gillenwater et al. (2021) for the multiple approximate quantiles problem by using a tree-based (recursive) approach to CDF estimation. However, none of these works deal with quantile estimation in the sublinear space setting. Böhler & Kerschbaum (2020) solve the problem of estimating the joint median of two private data sets with time complexity sub-linear in the size of the data-universe and provide privacy guarantees for small data sets as well as limited group privacy guarantees unconditionally against polynomially time-bounded adversaries.

Inherent privacy: Another line of work (Blocki et al., 2012; Smith et al., 2020; Choi et al., 2020) demonstrates that sketching algorithms for streaming problems might have inherent privacy guarantees under minimal assumptions on the dataset in some cases. For such algorithms, relatively little noise needs to be added to preserve privacy unconditionally.

3 Preliminaries and Notation

We now give standard differential privacy notation and formally describe the quantile estimation problem. We also present the Greenwald-Khanna sketch guarantees in a suitable form.

3.1 Differential Privacy

Definition 3.1 (Differential Privacy (Dwork et al., 2006))**.** Let $\mathcal{Q} : \mathcal{X}^n \to \mathcal{R}$ be a (randomized) mechanism. For any $\epsilon \geq 0, \delta \in [0, 1]$, $\mathcal{Q}$ satisfies (ϵ, δ)**-differential privacy** if for any neighboring databases (that differ in one row) $\mathbf{x} \sim \mathbf{x}' \in \mathcal{X}^n$ and any set $S \subseteq \mathcal{R}$,

$$\mathbb{P}[\mathcal{Q}(\mathbf{x}) \in S] \leq e^{\epsilon}\mathbb{P}[\mathcal{Q}(\mathbf{x}') \in S] + \delta.$$

The probability is taken over the coin tosses of $\mathcal{Q}$. We say that $\mathcal{Q}$ satisfies pure differential privacy (ϵ-DP) if $\delta = 0$ and approximate differential privacy ((ϵ, δ)-DP) if $\delta > 0$. We can set ϵ to be a small constant (e.g., between 0.01 and 2) but will require that $\delta \leq n^{-\omega(1)}$ be cryptographically small.

3.2 Quantile Approximation

In this subsection we make some definitions to formalize our analysis.

Definition 3.2. (Quantiles) Let there be a totally ordered data universe $\mathcal{X}$ and an input data stream $X = ((x_1, 1), \ldots, (x_n, n))$ (sometimes implicitly referred to as $X = (x_1, \ldots, x_n)$). For $(x_i, i) \in X$, let $\mathrm{val}((x_i, i)) = x_i$ and $\mathrm{ix}((x_i, i)) = \sum_{j \leq i} |\{(x_j, j) : x_j < x_i \text{ or } x_j = x_i, j < i\}|$. We abuse notation to say that for $v_1, v_2 \in X$, $v_1 \leq v_2$ if $\mathrm{ix}(v_1) \leq \mathrm{ix}(v_2)$. Then, the q-quantile of X is $\mathrm{val}(v)$ for $v \in X$ with $\mathrm{ix}(v) = \lceil qn \rceil$.

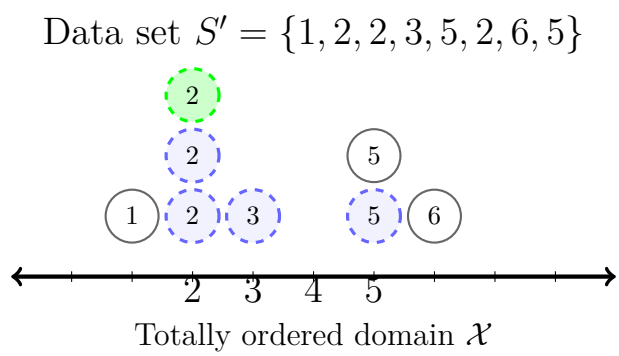


Figure 1: In the data set S' of 8 elements, the true 0.5 quantile is the value 2, and the values $2, 3, 4$ and 5 are all acceptable 0.25-approximate answers. Note that although 1 is adjacent to 2 in the data universe $\mathcal{X}$, it is not an acceptable output.

We observe that in this setting a value from the data universe can occur multiple times in a set. To account for this we define a range of ranks that any value can hold; this will be useful when reasoning about quantile approximation.

Definition 3.3. (Rank and approximate quantiles) For $x \in \mathcal{X}$, we define $r_{\min}(x) = |\{v \in X : \mathrm{val}(v) < x\}|$, $r_{\max}(x) = |\{v \in X : \mathrm{val}(v) \leq x\}|$ and $\mathrm{rank}(X, x)$ to be the interval $[r_{\min}(x), r_{\max}(x)]$. We say that $x \in \mathcal{X}$ is an α-approximate q-quantile for X if $\mathrm{rank}(X, x) \cap [\lceil qn \rceil - \alpha n, \lceil qn \rceil + \alpha n] \neq \emptyset$.

Consider the following example to see how these definitions play out in practice.

Example 3.4. Given the data set $\{1, 2, 2, 3, 5, 2, 6, 5\}$ (refer to figure 1), the 0.5 quantile is 2, which we distinguish from the median, which would be the average of the elements ranked 4 and 5, i.e, 2.5 for this data set. A straightforward way to obtain quantiles is to sort the dataset and the pick the element at the $\lceil q \cdot n \rceil$ position. This method only works in the offline (non-streaming) setting. For $\alpha = 0.25$, the α-approximate 0.5 quantiles are $2, 3, 4$ and 5. Note that 4, which does not occur in the data set, is still a valid response, but 1, which occurs in the data set and is even adjacent to the true 0.5 quantile 2 in the data universe $\mathcal{X}$, is not a valid response.

We can now formalize the two versions of the problems that are dealt with in the literature on streaming quantile approximation. Let $\mathcal{D}$ be any distribution with random variable $X \sim \mathcal{D}$ and CDF $F_X : \mathbb{R} \to [0, 1]$. For any $q \in [0, 1]$, $Q^q_{\mathcal{D}}$ denotes the value x such that $F_X(x) = q$.

Definition 3.5 (Single Quantile)**.** Given sample $S = (x_1, \ldots, x_n)$ in a streaming fashion in arbitrary order, construct a data structure for computing the quantile $Q^q_{\mathcal{D}}$ such that for any $q \in [0, 1]$, with probability at least $1 - \beta$, $|Q^q_{\mathcal{D}} - \tilde{Q}^q_S| \leq \alpha$.

Definition 3.6 (All Quantiles)**.** Given sample $S = (x_1, \ldots, x_n)$ in a streaming fashion in arbitrary order, construct a data structure for computing the quantile $Q^q_{\mathcal{D}}$ such that with probability at least $1 - \beta$, for all values of $q \in M$ where $M \subset [0, 1]$, $|Q^q_{\mathcal{D}} - \tilde{Q}^q_S| \leq \alpha$.

3.3 Non-private Quantile Streaming

We now state a formal generic quantile streaming algorithm definition that might be useful for extending our work:

Definition 3.7 (Sketching with Property $\mathcal{P}$)**.** Let S be a deterministic or randomized sketching algorithm that maps a stream X to a summary $S(X)$. Let $\alpha(S(X), X)$ denote the error associated with the summary $S(X)$ on the stream X. Then the sketching algorithm should satisfy $\alpha(\phi(S(X), X)) \leq \alpha(S(X), X)$ for all $\phi \in \mathcal{P}$.

We now specialize Definition 3.7 to the formal guarantees of the GK sketch:

Lemma 3.8. *[GK guarantees (Greenwald & Khanna, 2001)] Given a data stream $x_1, \ldots, x_n$ of elements drawn from $\mathcal{X}$, the GK sketch outputs a list $S(X)$ of $s(= O((1/\alpha)\log(\alpha n)))$ many tuples $(v_i, g_i, \Delta_i) \in (\mathcal{X}\times\mathbb{N})\times\mathbb{N}\times\mathbb{N}$ for $i = 1, \ldots, s$ such that if X is the data multi-set then: (1) $\mathrm{rank}(X, \mathrm{val}(v_i)) \subset [\sum_{j\leq i} g_j, \Delta_i + \sum_{j\leq i} g_j]$. (2) $g_i + \Delta_i \leq 2\alpha n$. (3) The first tuple is $(\min\{x \in X\}, 1, 0)$ and the last tuple is $(\max\{x \in X\}, 1, 0)$. (4) The v_i are sorted in ascending order. Without loss of generality, the lower interval bounds $\sum_{j\leq i} g_j$ and upper interval bounds $\Delta_i + \sum_{j\leq i} g_j$ are also sorted in increasing order.*

We reproduce the proof in appendix A.5.

4 Differentially Private Algorithms

In this section we present our two DP mechanisms for quantile estimation. Throughout, we assume that α is a user-defined approximation parameter. The goal is to obtain α-approximate q-quantiles.

The new algorithms we introduce are:

(1) `DPExpGK` (Algorithm 1): An exponential mechanism based $(\epsilon, 0)$-DP algorithm for computing a single q-quantile. To solve the all-quantiles problem with approximation factor α, one can run this algorithm iteratively with target quantile $0, \alpha, 2\alpha, \ldots$. Doing so requires scaling the privacy parameter in each call by an additional α factor which increases the space complexity by a factor of $1/\alpha$. We also give an optimized implementation (algorithm 8) of this algorithm in the appendix. We extend our results to the continual observation setting (Dwork et al., 2010; Chan et al., 2011).

(2) `DPHistGK` (Algorithm 2): A histogram based $(\epsilon, 0)$-DP algorithm (Algorithm 2) for the α-approximate all-quantiles problem. The privacy guarantee of this algorithm is unconditional, but there is no universal theoretical utility bound as in the previous algorithm. However, in some cases the utility is provably better: for example, we show that if the data set is drawn from a normal distribution (with unknown mean and variance), we can avoid the quadractic $1/\alpha^2$ factor in the sample complexity that we incur when using `DPExpGKGumb` for the same all-quantiles task.

The rest of this section contains the descriptions of the two algorithms; we relegate all proofs to the appendix.

4.1 `DPExpGKGumb`: Exponential Mechanism Based Approach

We first establish how the exponential mechanism and the GK sketch may be used in conjunction to solve the single quantile problem. Concretely, the high level idea is to call the privacy preserving exponential mechanism with a score function derived from the GK sketch. The exponential mechanism is a fundamental privacy primitive which when given a public set of choices and a private score for each choice outputs a choice that with high probability has a score close to optimal whilst preserving privacy. In the course of constructing our algorithms, we have to resolve two problems; one, how to usefully construct a score function to pass to the exponential mechanism so that the private value derived is a good approximation to the q-quantile, and two, how to execute the exponential mechanism efficiently on the (possibly massive) data universe $\mathcal{X}$. On resolving the first issue we get the (not necessarily efficient) routine Algorithm 1, and on resolving the second issue we get an essentially equivalent but far more efficient routine i.e. Algorithm 8.

Algorithm 1: `DPExpGK`: Exponential Mechanism DP Quantiles : High Level Description

Data: $X = (x_1, x_2, \ldots, x_n)$

Input: ϵ, α (approximation parameter), $q \in [0, 1]$ (quantile parameters), δ_u (sensitivity)

1 $S(X) = \{(v_i, g_i, \Delta_i) : i \in [s]\} \leftarrow GK(X, \alpha)$

2 Define score function

$$u(S(X), x) = -\min\{|y - \lceil qn \rceil| : y \in [\hat{r}_{\min}(x), \hat{r}_{\max}(x)]\}. \tag{1}$$

where

$$\hat{r}_{\min}(x) = \max\{\sum_{j \leq i} g_j : \mathrm{val}(v_i) < x\}$$
$$\hat{r}_{\max}(x) = \min\{\Delta_i + \sum_{j \leq i} g_j : \mathrm{val}(v_i) > x\}$$

3 Execute the exponential mechanism with score function $u(S(X), \cdot)$, i.e. choose and output a single $e \in \mathcal{X}$ with probability

$$\propto \exp\left(\frac{\epsilon}{2\delta_u} \cdot u(S(X), e)\right).$$

Constructing a score function: We recall that the GK sketch returns a short sequence of elements from the data set with a deterministic interval for their ranks and the promise that for any target quantile $q \in [0, 1]$, there is some sketch element that lies within αn units in rank of $\lceil qn \rceil$. One technicality that we run into when trying to construct a score function on the data universe $\mathcal{X}$ is that when a single value occurs with very high frequency in the data set, the ranks of the set of occurrences can span a large interval in $[0, n]$, and there is no one rank we can ascribe to it so as to compare it with the target rank $\lceil qn \rceil$. This can be resolved by defining the score for any data domain value in terms of the distance of its respective rank interval $[r_{\min}(x), r_{\max}(x)]$ (formalized in Definition B.1) from $\lceil qn \rceil$; elements whose intervals lie closer to the target have a higher score than further away ones.

Efficiently executing the exponential mechanism: The exponential mechanism samples one of the public choices (in our case some element from the data universe $\mathcal{X}$) with probability that increases with the quality of the choice according to the score function. In general every element can have a possibly different score and the efficiency of the exponential mechanism can vary widely depending on the context. In our setting, the succinctness of the GK sketch leads to a crucial observation: by defining the score function via the sketch, the data domain is partitioned into a relatively small number of sets such that the score function is constant on each partition. Concretely, for any two successive elements in the GK sketch, the range of values in the data universe that lie between them will have the same score according to our score function. We can hence first sample a partition from which to output a value, and then choose a value from within that interval uniformly at random. To make our implementation even more efficient and easy to use, we also make use of the *Gumbel-max* trick that allows us to iterate through the set of choices instead of storing them in memory - see appendix B.1 and the expanded pseudocode in algorithm 8 for more detail.

On formalizing this outline we get the following formal guarantees for algorithm 1 (please see appendix B.1 for a complete proof). $d(\cdot, \cdot)$ denotes the ℓ_1 metric on $\mathbb{R}$.

Theorem 4.1. *Algorithm 1 is ϵ-differentially private. Let $\hat{x}$ be the value returned by Algorithm 1 when initialized with target quantile q. The following statements hold:*

1. *Algorithm 1 can be implemented to run with space complexity $O((1/\alpha)\log \alpha n)$, such that with probability $1 - \beta$*

$$d(\lceil qn \rceil, [\hat{r}_{\min}(\hat{x}), \hat{r}_{\max}(\hat{x})]) \leq 2\alpha n + \frac{2(4\alpha n + 2)\log(|\mathcal{X}|/\beta)}{\epsilon} \tag{2}$$

2. *For* $n > \frac{\log|\mathcal{X}|/\beta}{12\alpha\epsilon}$, *Algorithm 1 can be implemented to run with space complexity* $O\left((\alpha\epsilon)^{-1}\log(|\mathcal{X}|/\beta)\log(\alpha\epsilon n)\right)$ *such that with probability* $1-\beta$, $\hat{x}$ *is an* α *approximate* q*-quantile.*

4.2 `DPHistGK`: Histogram Based Approach

For methods in this section, we assume that we have $K \geq 1$ *disjoint* bins each of width w (e.g., $w = \alpha/2$) partitioning the data universe. These bins are used to construct a histogram. Essentially, Algorithm 2 builds an empirical histogram based on the GK sketch, adds noise so that the bin values satisfy $(\epsilon, 0)$-DP, and converts this empirical histogram to an approximate empirical CDF, from which the quantiles can be approximately calculated. We demonstrate one use case of `DPHistGK` where the space complexity required improves upon the worst-case bound for `DPExpGK`, Theorem 4.1. While the histogram based mechanism does not have universal utility bounds in the spirit of the above theorem, the results in this section serve as one simple example where it may yield desirable accuracy while using less space.

Algorithm 2: `DPHistGK`: Computing DP Quantiles in Bounded Space

Data: $X = (x_1, x_2, \ldots, x_n)$
Input: ϵ, α (approximation parameter), $q \in [0,1]$ (quantile parameters), w

Build summary sketch $S(X)$ where
$S(X) = \{(v_i, g_i, \Delta_i) : i \in [s]\} \leftarrow GK(X, \alpha)$
`/* cell labels` a_i `and counts` $c_i = 0$ `*/`
Initialize data-agnostic (empty) histogram $Hist = \langle (a_i, c_i), \ldots \rangle$ with cell widths w
for $(v_i, g_i, \Delta_i) \in S(X)$ **do**
  Insert g_i counts of v_i into histogram $Hist$
$c = 0$
$H = [\cdot]$`/* initialize empty list */`
for $(a_i, c_i) \in Hist$ **do**
  $\tilde{c}_i = \max(0, c_i + \mathrm{Lap}(0, 2/\epsilon))$
  Append $(a_i, c + \tilde{c}_i)$ to H
  $c = c + \tilde{c}_i$
$r = \lceil q \cdot n \rceil$
for $(b, rank) \in H$ **do**
  if $r < rank$ **then**
    return b
`/* return last element of` H `*/`
return $H[|H| - 1]$

Suppose that we are given an i.i.d. sample $S = (X_1, X_2, \ldots, X_n)$ such that for all $i \in [n]$, $X_i \sim \mathcal{N}(\mu, \sigma^2 I_{d\times d})$, $\mu \in \mathbb{R}^d$. The goal is to estimate DP quantiles of the distribution $\mathcal{N}(\mu, \sigma^2 I_{d\times d})$ without knowledge of μ or σ^2. We will show how to estimate the quantiles assuming that σ^2 is known. Note that it is easy to generalize the work to the case where σ^2 is unknown as follows:

For any sample S drawn from i.i.d. from $\mathcal{N}(\mu, \sigma^2 I_{d\times d})$, the $1-\beta$ confidence interval is

$$\bar{X} \pm \frac{\sigma}{\sqrt{n}} \cdot z_{1-\beta/2},$$

where $z_{1-\beta/2}$ is the $1-\beta/2$ quantile of the standard normal distribution and $\bar{X}$ is the empirical mean. The length of this interval is fixed and equal to $\frac{2\sigma z_{1-\beta/2}}{\sqrt{n}} = \Theta\left(\frac{\sigma}{\sqrt{n}}\sqrt{\log\frac{1}{\beta}}\right)$. In the case where σ^2 is unknown, the confidence interval becomes $\bar{X} \pm \frac{s}{\sqrt{n}} \cdot t_{n-1,1-\beta/2}$, where $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ is the sample variance (sample estimate of σ^2) and $t_{n-1,1-\beta/2}$ is the $1-\beta/2$ quantile of the t-distribution with $n-1$ degrees of freedom. The length of the interval can be shown to be

$$\frac{2\sigma}{\sqrt{n}} \cdot k_n \cdot t_{n-1,1-\beta/2} = \Theta\left(\frac{\sigma}{\sqrt{n}}\sqrt{\log\frac{1}{\beta}}\right),$$

where $k_n = 1 - O(1/n)$ is an appropriately chosen constant (see (Lehmann & Romano, 2005; Karwa & Vadhan, 2018; Keener, 2010) for more details and discussion). We can hence assume that σ^2 is known and proceed to show sample and space complexity bounds. [2]

For any $q \in (0,1)$, we denote the q-quantile of the sample $S = (X_1, X_2, \ldots, X_n)$ as Q_S^q and the q-quantile of the distribution as $Q_{\mathcal{D}}^q$. With this notation, for some $\beta \in (0,1]$ and $\alpha > 0$, we wish to obtain a DP q-quantile $\tilde{Q}_S^q$ such that

$$\mathbb{P}[\|Q_{\mathcal{D}}^q - \tilde{Q}_S^q\| \geq \alpha] \leq \beta.$$

We shall proceed in a three-step approach: (1) Estimate a DP range of the population in sub-linear space; (2) Use this range of the population to construct a DP histogram using the stream S; (3) Use the DP histogram to estimate one or more quantiles via the sub-linear data structure of Greenwald and Khanna. Our main formal result for algorithm 2 can be summarized as follows (please see appendix B.2 for a complete proof).

Theorem 4.2. *For the one-dimensional normal distribution $\mathcal{N}(\mu, \sigma^2)$, let $S = (X_1, X_2, \ldots, X_n)$ be a data stream through which we wish to obtain $\tilde{Q}_S^q$, a DP estimate of the q-quantile of the distribution.*

For any $q \in (0,1)$, there exists an (ϵ, δ)-DP algorithm such that, with probability at least $1 - \beta$, we obtain $|Q_{\mathcal{D}}^q - \tilde{Q}_S^q| \leq \alpha$ for any $\alpha > 0$, $\beta \in (0,1], \epsilon, \delta \in (0, 1/n)$ and for stream length

$$n \geq \max\left\{\min\left\{A, B\right\}, C\right\}, \quad \textit{where}$$

$$A = O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{R}{\sigma\beta}\right), B = O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{1}{\beta\delta}\right), C = O\left(\frac{R^2}{\sigma^2\alpha^2}\log\frac{1}{\beta}\right)$$

as long as $\mu \in (-R, R)$ and using space of $O(\max\{\frac{R}{\sigma}, \frac{1}{\alpha}\log\alpha n\})$.

In the case where $d = 1$, by Theorem 4.2, there exists an (ϵ, δ)-DP algorithm $\tilde{Q}_S^q$ such that if $\mu \in (-R, R)$ then using space of $O(\max\{\frac{R}{\sigma}, \frac{1}{\alpha}\log\alpha n\})$ (with probability 1) as long as the stream length n is at least

$$\Omega\left(\max\left\{\min\left\{O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{R}{\sigma\beta}\right), O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{1}{\beta\delta}\right)\right\}, O\left(\frac{R^2}{\sigma^2\alpha^2}\log\frac{1}{\beta}\right)\right\}\right),$$

we get the guarantee that, for all $\beta \in (0,1], \alpha > 0$, $\mathbb{P}[\|Q_{\mathcal{D}}^q - \tilde{Q}_S^q\| \geq \alpha] \leq \beta$. Intuitively, this means that: (1) **Space**: We need less space to estimate any quantile with DP guarantees if the distribution is less concentrated (i.e., σ can be large) or if we do not require a high degree of accuracy for our queries (i.e., α can be large). (2) **Stream Length**: We need a large stream length to estimate quantiles if we require a high degree of accuracy (i.e., smaller β, α), or do not have a good public estimate of μ (large R), or have small privacy parameters (small ϵ, δ), or have concentrated datasets (small σ).

5 Continual Observation

We now describe how our one-shot approach can be used as a black box to obtain a continual observation solution (Dwork et al., 2010; Chan et al., 2011). In the absence of privacy, we recall that an $\alpha/2$-approximate solution for the q-quantile problem on a stream of length n allows for an additive error of $\alpha n/2$ in the rank of any candidate q-quantile solution. It follows that if we append any arbitrary $\alpha n/2$-many elements to the end of the stream, these new elements can lead to an additional additive error of at most $\alpha n/2$. This suggests that on releasing a DP quantile estimate at any point in the stream, we can avoid updating our q-quantile estimate for a proportionate number of additional elements whilst maintaining a $(1+\alpha)$-approximation. This is essentially the idea behind the *flip number* used in the study of adversarially robust streaming algorithms. At a high level, the flip number is a measure of the number of times the output of an online algorithm changes by more than a $(1 \pm \alpha)$ multiplicative factor (see Ben-Eliezer et al. (2020) for more detail).

Building on this idea, we can show that we only need update our estimate after every additional $\alpha/2$-fraction as many elements are added as were present at the previous estimate. At the beginning of this process, i.e. when no stream elements have been processed, for the first few elements ($n_{\min}$-many, as described in

[2] One could also estimate the variance in a DP way and then prove the complexity bounds.

Algorithm 3: Continual Observation DP Quantiles

Data: Input stream $X = (x_1, \ldots, x_n)$ for some $n > n_{\min} = \Omega\left(\frac{1}{\epsilon\alpha^2}\log^2 n \log\left(\frac{|\mathcal{X}|\log n}{\alpha\beta}\right)\right)$, privacy parameter ϵ, approximation parameter α, target quantile q

, failure probability β Let $s = 0$

$\epsilon^* \leftarrow O(\alpha\epsilon/\log n)$

$\alpha^* \leftarrow \alpha/2$

$\beta^* \leftarrow \frac{\log n}{\alpha\beta}$

$\mathrm{cp}(s) \leftarrow n_{\min}$ `/* stores stream checkpoints */`

Instantiate GK $\leftarrow GK(\alpha^*)$

$v_s \leftarrow \perp$ `/* holds` α`-approximate` q`-quantile */`

for *stream element* $x_s \in X$ **do**

  $s \leftarrow s + 1$

  GK.$insert(x_s)$

  if $s = \lceil \mathrm{cp}(s-1) \rceil$ **then**

    $v_s \leftarrow$ `DPExpGK`$(\mathrm{GK}, \epsilon^*, \alpha^*, q, \beta^*)$

    $\mathrm{cp}(s) \leftarrow \mathrm{cp}(s-1)(1 + \alpha/2)$

  else

    $v_s \leftarrow v_{s-1}$

    $\mathrm{cp}(s) \leftarrow \mathrm{cp}(s-1)$

the pseudo-code) we will need to update our online estimate every time or omit producing any estimate. However, after this short warm-up prefix of the stream, a relatively small number of estimates serve as $(1+\alpha)$-approximate q quantiles for every point in the remainder of the stream.

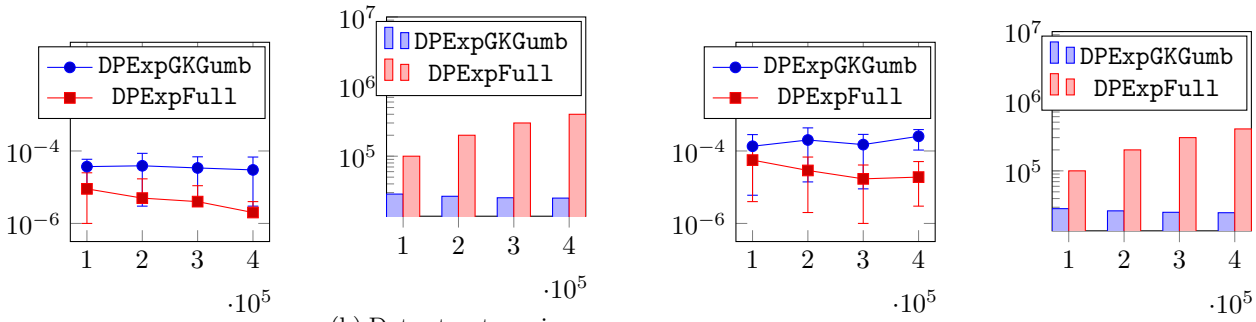
Accounting for the additional requirement of privacy is now easy - we merely need to privatize the estimate at each of the *checkpoints*, which are the points in the stream where the quantile estimate must be updated. For n elements in all there are $O\big(\log_{1+\alpha}\frac{n}{n_{\min}}\big)$ many such elements, which for a suitable choice of n_{min} simplifies essentially to $O\big(\frac{\log n}{\epsilon\alpha}\big)$. In other words we see that the privacy loss scales only with the logarithm of the length of the stream because of the relatively few checkpoints that occur. The formal statement is as below, and we present a complete proof in appendix B.3.

Theorem 5.1. *Let $\epsilon, \alpha > 0$, $n \in \mathbb{Z}$. For any $\beta \in (0, 1]$, with probability $\geq 1 - \beta$, Algorithm 3 maintains an α-approximate q-quantile at every point s in the data stream such that $s = \Omega\left(\frac{1}{\epsilon\alpha^2}\log^2 n \log\left(\frac{|\mathcal{X}|\log n}{\alpha\beta}\right)\right)$. Furthermore, Algorithm 3 satisfies ϵ-DP.*

We conclude by mentioning that our continual observation solution incurs roughly a $O(\log n/\alpha)$ overhead in the space complexity, which is in line with classical works in differential privacy and adversarially robust streaming (Ben-Eliezer et al., 2020; Dwork et al., 2010). Concurrently, Stemmer and Kaplan (Kaplan & Stemmer, 2021) developed a notion of *streaming sanitizers* which yields a continual observation guarantee "for free", without incurring such an overhead over the one-shot case.

6 Experimental Evaluation

In this section, we experimentally evaluate our sublinear-space exponential-based mechanism, `DPExpGKGumb`. We study how well our algorithm performs in terms of accuracy and space usage. This section validates the theoretical results in the prequel - we find that the space complexity of the algorithm is indeed very small in practice, and that the accuracy is typically closely tied to the approximation parameter α. Our main baselines will be `DPExpFull`, which applies the exponential mechanism on the full data set without any sketching algorithms (see appendix C.1 for further implementation details), and the true quantile value. The true quantile values are used to compute *relative error*, the absolute value of the difference between the estimated quantile and the true quantile, divided by the standard deviation of the data set. We graph the mean relative error over 100 trials of the exponential mechanism per experiment, as well as the 80%



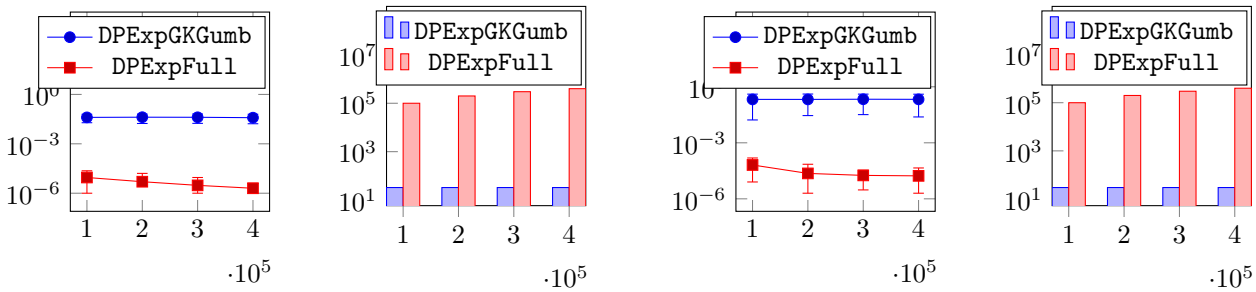
(a) Error vs. stream length for Uniform distribution.

(b) Data structure sizes vs. stream length for Uniform distribution.

(c) Error vs. stream length for Normal distribution.

(d) Data structure sizes vs. stream length for Normal distribution.

Figure 2: `DPExpGKGumb` versus `DPExpFull` for uniform or normally distributed data, $\alpha = 10^{-4}$, $\epsilon = 1$, $q = 0.5$



(a) Error vs. stream length for Uniform distribution

(b) Data structure sizes vs. stream length

(c) Error vs. stream length for Normal distribution

(d) Data structure sizes vs. stream length

Figure 3: `DPExpGKGumb` versus `DPExpFull` for uniform or normally distributed data, $\alpha = 10^{-1}$, $\epsilon = 1$, $q = 0.5$

confidence interval computed by taking the 10th and 90th percentile. We fix the target quantile to be estimated to $q = 0.5$ (the median) throughout as the relative error and the space usage generally do not seem to vary much with choice of q.

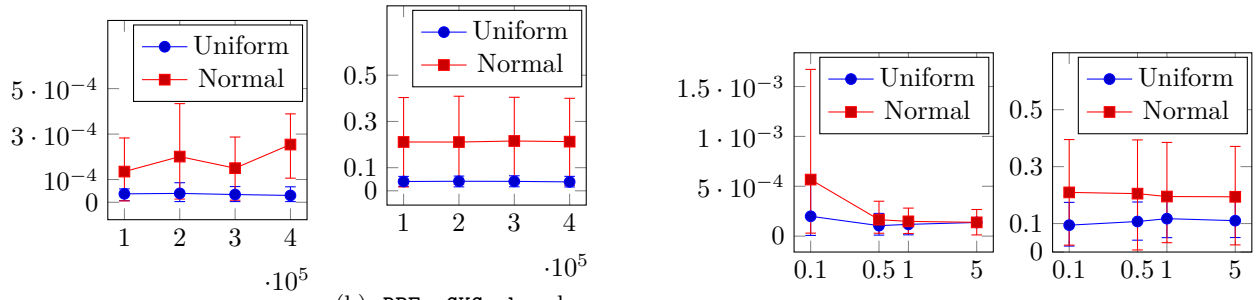
6.1 Synthetic data sets

In this subsection, we compare our methods on synthetically generated datasets. We vary the parameters of the **Stream Length** n (the size of the input stream $x_1, \ldots, x_n$), the **Approximation Parameter** α (the approximation factor used by the internal GK sketch), and the **Data Distribution**. We generate data from uniform and Gaussian distributions; we use a uniform distribution in range $[0, 1]$ (i.e., $U(0, 1)$) or a normal distribution with mean 0 and variance 1 (i.e., $N(0, 1)$), clipped to the interval $[-10, 10]$.[3].

We show results on space usage and relative error (both plotted mostly on logarithmic scales) from non-DP estimates, as we vary the parameters listed above.

In Figure 2, we vary the stream length n for an approximation factor of $\alpha = 10^{-4}$. The streams are either normally or uniformly distributed. In Figures 2a and 2b, we compare `DPExpGKGumb` (space strongly sublinear in n) vs. `DPExpFull` (uses space of $O(n)$) in terms of space usage and accuracy. In general we find that although our method `DPExpGKGumb` incurs higher error, in absolute terms it remains quite small and the 95%

[3]Clipping is required for the exponential mechanism, as it must operate on some bounded interval of values. In any case, we never expect to see samples from $N(0, 1)$ that lie outside $[-10, 10]$ for any practical purpose; the probability for any given sample to satisfy this is minuscule, at about $\approx 10^{-21}$.



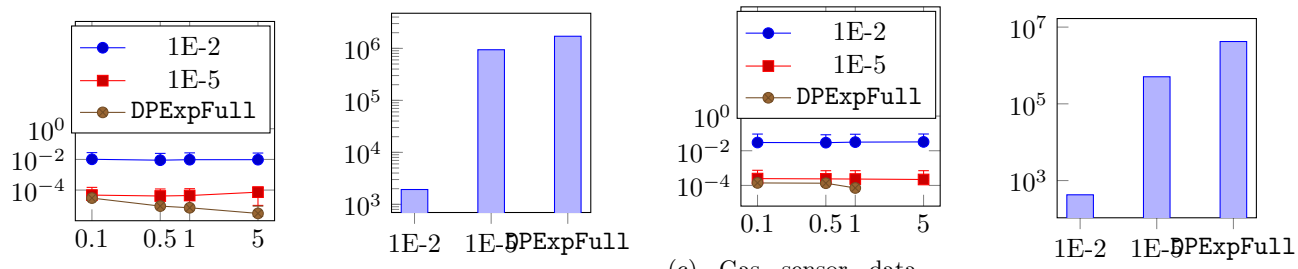
(a) Relative error vs. datastream size for $\alpha = 10^{-4}$, $\epsilon = 1$, $q = 0.5$

(b) `DPExpGKGumb` relative error vs. datastream size for $\alpha = 10^{-1}$, $\epsilon = 1$, $q = 0.5$

(c) Approx. factor $\alpha = 10^{-4}$

(d) Approx. factor $\alpha = 10^{-1}$

Figure 4: Relative error versus approximation factor.



(a) Taxi data set: Error versus privacy parameter ϵ for `DPExpGKGumb` for approximation $\alpha = 10^{-2}$ and 10^{-5}, and `DPExpFull`.

(b) Taxi data set: Space used by `DPExpGKGumb` for approximation $\alpha = 10^{-2}$ and 10^{-5}, and `DPExpFull`.

(c) Gas sensor data set: Error versus privacy parameter ϵ for `DPExpGKGumb` for approximation $\alpha = 10^{-2}$ and 10^{-5}, and `DPExpFull`[4].

(d) Gas sensor data set: Space used by `DPExpGKGumb` for approximation $\alpha = 10^{-2}$ and 10^{-5}, and `DPExpFull`.

Figure 5: Taxi and Gas sensor data sets.

confidence intervals tend to be adjacent for `DPExpGKGumb` and `DPExpFull`. However, there is a clear trend of an exponential gap developing between their respective space usages which is a natural consequence of the space complexity guarantee of the GK sketch. This holds for both distributions studied.

In Figure 3, we vary the stream length for a relatively large approximation factor of 0.1. Here we see that compared to the non-approximate method we incur far higher error, although there is also a concomitant increase in the space savings. This is not a typical use-case since the non-private error can itself be large, but we get a complete picture of how space usage and performance vary with this user-defined parameter.

In Figures 4c and 4d, we vary the approximation factor. In the small approximation setting we see the inverse tendency of accuracy with privacy which is characteristic of most DP algorithms. However, in the large approximation setting, there is no such clear drop in performance with more privacy. This motivates the question of determining the true interplay between the approximation factor α and the private parameter ϵ, as discussed further in Section 7.

Our algorithm performs well in practical settings where one wishes to estimate some quantity across all data items privately and using small space. For example, our results indicate that choosing an approximation factor of $\alpha = 10^{-4}$ induces an error which is also of order about 10^{-4} for privately computing parameters chosen according to a uniform or normal distribution, all while saving orders of magnitude in the space complexity.

6.2 Real-World Datasets

We repeat our investigation of the utility and space complexity comparison between `DPExpGKGumb` and `DPExpFull` with the following real-world data sets (Dua & Graff, 2017): (1) **Taxi Service Trajectory**: A dataset from the UCI machine learning repository describing trajectories performed by all 442 taxis (at the time) in the city of Porto in Portugal (Moreira-Matias et al., 2013). This dataset contains real-valued attributes with about 1.5 million instances. (2) **Gas Sensor Dataset**: A UCI repository dataset containing recordings of 16 chemical sensors exposed to varying concentrations of two gas mixtures (Fonollosa et al., 2015). The sensor measurements are acquired continuously during a 12-hour time range and contains about 4 millions instances. We pick a real-valued attribute from each dataset (the TIMESTAMP and the first ETHYLENE_CO gas sensor value, respectively) and calculate the median on these datasets. The results are reported in Figures 5a and 5b, and Figures 5c and 5d respectively.

We see that the larger the approximation factor, the larger the space savings are with `DPExpGK`. Comparing `DPExpFull` to `DPExpGK`, we see space savings of 2 times up to 1000 times as we vary the approximation factor. These results are consistent with our expectations that the space savings are inversely proportional to the allowed approximation factor.

7 Conclusion & Future Work

In this work, we presented sublinear-space and differentially private algorithms for approximately estimating quantiles in a dataset. Our solutions are two-part: one based on the exponential mechanism and efficiently implemented via the use of the Gumbel distribution; the other based on constructing histograms. Our algorithms are supplemented with theoretical utility guarantees. Furthermore, we experimentally validate our methods on both synthetic and real-world datasets. Our work leaves room for further exploration in various directions.

Interplay between α and ϵ: The space complexity bounds we obtain are (up to lower order terms) inversely linear in α and in ϵ. While it is either known or easy to show that such linear dependence in each of these parameters in itself is necessary, it is not clear whether the $\alpha^{-1}\epsilon^{-1}$ term in Theorem 4.1 can be replaced with, say, $\alpha^{-1} + \epsilon^{-1}$. Such an improvement, if possible, seems to require substantially modifying the baseline Greenwald-Khanna sketch or adding randomness.

Alternative streaming baselines: We base our mechanisms upon the GK-sketch, which is known to be space-optimal among deterministic streaming algorithms for quantile approximation. The use of a deterministic baseline simplifies the analysis and the overall solution, but better randomized streaming algorithms for the same problem are known to exist. What would be the benefit of working, e.g., with the (optimal among randomized algorithms) KLL-sketch (Karnin et al., 2016)?

Dependence in universe size: The dependence of our space complexity bounds in the size of the universe, $\mathcal{X}$, is logarithmic. Recent work of Kaplan et al. (Kaplan et al., 2020) (see also (Bun et al., 2015)) on the sample (not space) complexity of privately learning thresholds in one dimension, a fundamental problem at the intersection of learning theory and privacy, demonstrate a bound polynomial in $\log^* |\mathcal{X}|$ on the sample complexity. As quantile estimation and threshold learning are closely related problems, this raises the question of whether techniques developed in the aforementioned papers can improve the dependence on $|\mathcal{X}|$ in our bounds.

Random order: The results presented here (except for those about normally distributed data) all assume that the data stream is presented in worst case order, an assumption that may be too strong for some scenarios. Can improved bounds be proved when the data elements are chosen in advance but their order is chosen randomly? This can serve as a middle ground between the most general case (which we address in this paper) and the case where data is assumed to be generated according to a certain distribution.

[4]The 80% CI for error incurred by `DPExpFull` was entirely supported on the point 0 and drops off axis as we use the log scale.

References

Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. *Perturbation Techniques in Online Learning and Optimization*, pp. 233–264. 2017. 23

Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26:1–26:28, 2013. 2, 3

Hilal Asi and John C. Duchi. Near instance-optimality in differential privacy. *CoRR*, abs/2005.10630, 2020. 1, 3

Victor Balcer and Salil P. Vadhan. Differential privacy on finite computers. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pp. 43:1–43:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 23

Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pp. 63–80, 2020. 8, 9

Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pp. 410–419. IEEE Computer Society, 2012. 3

Jonas Böhler and Florian Kerschbaum. Secure sublinear time differentially private median computation. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*, 2020. URL https://www.ndss-symposium.org/ndss-paper/secure-sublinear-time-differentially-private-median-computation/. 3

Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pp. 634–649, 2015. URL https://doi.org/10.1109/FOCS.2015.45. 12, 25

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011. 5, 8

Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. *Proc. Priv. Enhancing Technol.*, 2020(3):153–174, 2020. 3

Graham Cormode and Pavel Veselý. A tight lower bound for comparison-based quantile summaries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, pp. 81–93, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371087. doi: 10.1145/3375395.3387650. URL https://doi.org/10.1145/3375395.3387650. 3

Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pp. 263–272, 2006. 3

Graham Cormode, Zohar S. Karnin, Edo Liberty, Justin Thaler, and Pavel Veselý. Relative error streaming quantiles. *CoRR*, abs/2004.01668, 2020. URL https://arxiv.org/abs/2004.01668. 3

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. 12

A. Dvoretzky, J. Kiefer, and J. Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Ann. Math. Statist.*, 27(3):642–669, 09 1956. doi: 10.1214/aoms/1177728174. URL https://doi.org/10.1214/aoms/1177728174. 24

Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 371–380, 2009. 1, 3

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pp. 265–284, 2006. 4, 26

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pp. 715–724. ACM, 2010. 5, 8, 9

David Felber and Rafail Ostrovsky. A randomized online quantile summary in $o((1/\epsilon)\log(1/\epsilon))$ words. *Theory Comput.*, 13(1):1–17, 2017. 2, 3

Jordi Fonollosa, Sadique Sheik, Ramón Huerta, and Santiago Marco. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, 2015. ISSN 0925-4005. URL https://www.sciencedirect.com/science/article/pii/S0925400515003524. 12

Jennifer Gillenwater, Matthew Joseph, and Alex Kulesza. Differentially private quantiles. *CoRR*, abs/2102.08244, 2021. URL https://arxiv.org/abs/2102.08244. 1, 3

Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, pp. 58–66, 2001. URL https://doi.org/10.1145/375663.375670. 2, 3, 5, 17, 19

Michael Greenwald and Sanjeev Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*, pp. 275–285, 2004. URL https://doi.org/10.1145/1055558.1055597. 25

Michael B. Greenwald and Sanjeev Khanna. Quantiles and equi-depth histograms over streams. In *Data Stream Management - Processing High-Speed Data Streams*, pp. 45–86. 2016. 2

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1): 73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732. 1

Regant Y. S. Hung and Hing-Fung Ting. An $\omega(1/\epsilon\log(1/\epsilon))$ space lower bound for finding *epsilon*-approximate quantiles in a data stream. In *Frontiers in Algorithmics, 4th International Workshop, FAW 2010, Wuhan, China, August 11-13, 2010. Proceedings*, volume 6213 of *Lecture Notes in Computer Science*, pp. 89–100. Springer, 2010. doi: 10.1007/978-3-642-14553-7_11. URL https://doi.org/10.1007/978-3-642-14553-7_11. 2, 3

Haim Kaplan and Uri Stemmer. A note on sanitizing streams with differential privacy, 2021. 9, 24

Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2263–2285. PMLR, 2020. URL http://proceedings.mlr.press/v125/kaplan20a.html. 12

Haim Kaplan, Shachar Schnapp, and Uri Stemmer. Differentially private approximate quantiles. *CoRR*, abs/2110.05429, 2021. URL https://arxiv.org/abs/2110.05429. 3

Zohar S. Karnin, Kevin J. Lang, and Edo Liberty. Optimal quantile approximation in streams. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 71–78, 2016. 2, 3, 12

Vishesh Karwa and Salil Vadhan. Finite Sample Differentially Private Confidence Intervals. In Anna R. Karlin (ed.), *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 44:1–44:9, 2018. 8, 25

R.W. Keener. *Theoretical Statistics: Topics for a Core Course*. Springer Texts in Statistics. Springer New York, 2010. ISBN 9780387938394. 8

William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. doi: 10.1080/01621459.1952.10483441. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441. 1

Erich L. Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses*. Springer Texts in Statistics. Springer New York, New York, NY, 3. edition, 2005. ISBN 9780387276052. 8

Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *VLDB J.*, 25(4):449–472, 2016. doi: 10.1007/s00778-016-0424-7. URL https://doi.org/10.1007/s00778-016-0424-7. 3

Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pp. 251–262, 1999. URL https://doi.org/10.1145/304182.304204. 2, 3

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pp. 94–103, 2007. URL https://doi.org/10.1109/FOCS.2007.41. 20, 21

Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pp. 37–48, 2011. URL https://doi.org/10.1145/1989284.1989290. 2

Ilya Mironov. On significance of the least significant bits for differential privacy. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pp. 650–661, 2012. 23

Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luís Damas. Predicting taxi-passenger demand using streaming data. *IEEE Trans. Intell. Transp. Syst.*, 14(3):1393–1402, 2013. URL https://doi.org/10.1109/TITS.2013.2262376. 12

J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315–323, 1980. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(80)90061-4. URL https://www.sciencedirect.com/science/article/pii/0304397580900614. 2

Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 75–84, 2007. URL https://doi.org/10.1145/1250790.1250803. 1, 3

Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pp. 239–249, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138792. doi: 10.1145/1031495.1031524. URL https://doi.org/10.1145/1031495.1031524. 2

Adam D. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 813–822, 2011. URL https://doi.org/10.1145/1993636.1993743. 2, 21

Adam D. Smith, Shuang Song, and Abhradeep Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. In *NeurIPS*, 2020. 3

J. W. Tukey. A survey of sampling from contaminated distributions. *Contributions to Probability and Statistics*, pp. 448–485, 1960. URL https://ci.nii.ac.jp/naid/20000755025/en/. 1

Christos Tzamos, Emmanouil-Vasileios Vlatakis-Gkaragkounis, and Ilias Zadik. Optimal private median estimation under minimal distributional assumptions. In *NeurIPS*, 2020. 1, 3

Salil P. Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pp. 347–450. 2017. 25

Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. Quantiles over data streams: an experimental study. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pp. 737–748, 2013. 2

Zhuolun Xiang, Bolin Ding, Xi He, and Jingren Zhou. Linear and range counting under metric-based local differential privacy. In *IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, CA, USA, June 21-26, 2020*, pp. 908–913, 2020. URL https://doi.org/10.1109/ISIT44484.2020.9173952. 2

A Greenwald-Khanna Sketch

For completeness of our algorithm's description, we specify the operations in the Greenwald-Khanna (GK) non-private sketch. Throughout, we will use $n = n(t)$ to denote the number of elements encountered up to time $t \in \mathbb{Z}_+$. Some of the operations outlined here will be used a subroutines for the DP procedures.

A.1 The Sketch

Let $X = (x_1, x_2, \ldots, x_n)$ be a stream of items and $S(X)$ be the resulting sketch with size sublinear in n. The GK sketch stores

$$S(X) = \langle t_0, t_1, \ldots, t_{s-1} \rangle, \quad \forall i \in \{0, \ldots, s-1\}, t_i = (v_i, g_i, \Delta_i),$$

where $g_i = r_{min}(v_i) - r_{min}(v_{i-1})$ and $\Delta_i = r_{max}(v_i) - r_{min}(v_i)$. We reserve v_0, v_{s-1} be denote the smallest and largest elements seem in the stream X, respectively. We use $S(X)[i]$ to refer to the ith tuple in the sketch $S(X)$. i.e., for any i, $S(X)[i] = t_i = (v_i, g_i, \Delta_i)$.

Implicitly, the goal is to (implicitly) maintain bounds $r_{min}(v)$ and $r_{max}(v)$ for every v in $S(X)$. $r_{min}(v)$ and $r_{max}(v)$ are the lower and upper bounds on the rank of v amongst all items in X, respectively. We can compute these bounds as follows:

$$r_{min}(v_i) = \sum_{j \le i} g_j, \quad r_{min}(v_i) = \sum_{j \le i} g_j + \Delta_i.$$

As a result, $g_i + \Delta_i - 1$ is an upper bound on the number of items between v_{i-1} and v_i. In addition, $n = \sum_i g_i$.

The sketch is built in such a way to guarantee (maximum) error of $\max_{i=0}^{s-1}(g_i + \Delta_i)/2$ for approximately computing any quantile using the sketch.

We will also impose a tree structure over tuples in $S(X)$ (mostly because of the merge procedure) as follows: the tree $T(X)$ associated with $S(X)$ has a node V_i for each t_i. The parent of a node V_i is the node V_j such that j is the smallest index greater than i with $\text{band}(t_j) > \text{band}(t_i)$.

$\text{band}(t_i)$ is the band of Δ_i at time n and $\text{band}_\tau(n)$ as all tuples that had band value of τ. All possible values of Δ are denoted as bands and it can take on values between
$(0, \frac{1}{2}2\alpha n, \frac{3}{4}2\alpha n, \ldots, \frac{2^i-1}{2^i}2\alpha n, \ldots, 2\alpha n - 1, 2\alpha n)$ corresponding to capacities of $(2\alpha n, \alpha n, \ldots, 8, 4, 2, 1)$.

A.2 Quantile

Algorithm 4 computes the α-approximate q-quantile based on the sketch $S(X)$ that has size that is sublinear in n.

The algorithm goes through all tuples and checks if the condition $\max(r - r_{min}(v_i), r_{max}(v_i) - r) \leq \alpha n$ is satisfied and return (i, v_i) as the representative approximate quantile. This algorithm will be used a subroutine for one or more of our differentially private algorithms.

Lemma A.1 (Proposition 1 & Corollary 1 (Greenwald & Khanna, 2001))**.** *If after receiving n items in the stream, the sketch $S(X)$ satisfies the property $\max_i(g_i + \Delta_i) \leq 2\alpha n$, then Algorithm 4 returns an α-approximate q-quantile.*

Proof. The algorithm computes $r = \lceil qn \rceil$. Then the condition $\max(r - r_{min}(v_i), r_{max}(v_i) - r) \leq \alpha n$ clearly is (by definition) an α-approximate q-quantile. We still need to show that such v_i always exists. First set $e = \max_i(g_i + \Delta_i)/2$. If $r > n - e$, then $r_{min}(v_{s-1}) = r_{max}(v_{s-1}) = n$ so that $i = s - 1$ satisfies the property. When $r \leq n - e$, then the algorithm chooses the smallest index j such that $r_{max}(v_j) > r + e$ so that $r - e \leq r_{min}(v_{j-1})$. This follows since if $r - e > r_{min}(v_{j-1})$ then $r_{max}(v_j) = r_{min}(v_{j-1}) + g_j + \Delta_j > r_{min}(v_{j-1}) + 2e$ which contradicts the definition of e. □

Algorithm 4: Quantile$(S(X), q, n, \alpha)$: Computing α-Approximate Quantiles

Input: $S(X), q, n, \alpha$ (approximation parameter)

```
Compute r = ⌈qn⌉
for i = 0, ..., s − 1 do
    (v_i, g_i, Δ_i) = S(X)[i]
    if max(r − r_min(v_i), r_max(v_i) − r) ≤ αn then
        return (i, v_i)
return ⊥
```

A.3 Insert

Algorithm 5 goes through a stream of items and inserts into the sketch. The algorithm calls a compress operator on the data structure every time that $i \equiv 0 \mod \frac{1}{2\alpha}$ for any $i \in [m]$.

Algorithm 5: Inserting a stream of items into Summary Sketch

Data: $x_1, x_2, \ldots, x_m, \ldots$

Input: $S(X), \alpha$ (approximation parameter)

```
n = 0
for i = 1, ..., m, ... do
    if i ≡ 0 mod 1/(2α) then
        Compress(S(X), α, n)
    Insert(S(X), α, x_i)
    n = n + 1
return S(X), n
```

Algorithm 6 inserts a particular item x_n into the data structure $S(X)$. In the special case where x_n is a minimum or maximum, it inserts the tuple $(x_n, 1, 0)$ at the beginning or end of $S(X)$. Otherwise, it finds an index i such that $v_{i-1} \leq x_n < v_i$ and then inserts the tuple $(x_n, 1, \lfloor 2\alpha n \rfloor)$ into $S(X)$ at position i.

Algorithm 6: Insert$(S(X), \alpha, x_n)$: Inserting into Summary Sketch

Data: x_n
Input: $S(X), \alpha$ (approximation parameter)

```
(v_0, g_0, Δ_0) = S(X)[0]
(v_{s-1}, g_{s-1}, Δ_{s-1}) = S(X)[s-1]
if x_n < v_0 then
    Shift all positions in S(X)[0...s-1] to S(X)[i...s]
    S(X)[0] = (x_n, 1, 0)
else if x_n > v_{s-1} then
    S(X)[s] = (x_n, 1, 0)
else
    for i = 0, ..., s-1 do
        (v_i, g_i, Δ_i) = S(X)[i]
        if v_{i-1} ≤ x_n < v_i then
            Shift all positions in S(X)[i...s-1] to S(X)[i+1...s]
            S(X)[i] = (x_n, 1, ⌊2αn⌋)
return S(X), s+1
```

Algorithm 7: Compressing the Sketch

Input: $S(X), \alpha$ (approximation parameter), n

```
if n < 1/(2α) then
    return
for i = s-2, ..., 0 do
    t_i = (v_i, g_i, Δ_i) = S(X)[i]
    t_{i+1} = (v_{i+1}, g_{i+1}, Δ_{i+1}) = S(X)[i+1]
    Compute g*_i, the sum of g-values of tuple t_i and its descendants
    if band(t_i) ≤ band(t_{i+1}) & (g*_i + g_{i+1} + Δ_{i+1} < 2αn) then
        Delete all descendants of t_i and the tuple t_i from sketch S(X)
        Update t_{i+1} in S(X) to (v_{i+1}, g*_i + g_{i+1}, Δ_{i+1})
```

A.4 Compress

The Compress operation is an internal operation used for compressing (contiguous) tuples in $S(X)$. The goal of this operation is to merge a node and its descendants into either its right sibling or parent node. After merge, we have to maintain the property that the tuple is not full. A tuple is full when $g_i + \Delta_i \geq \lfloor 2\alpha n \rfloor$. By Proposition A.2, a node and its children will form a contiguous segment. Let g_i^* be the sum of g-values of tuple t_i and all of its descendants. Then merging t_i and its descendants would update t_{i+1} in $S(X)$ to $(v_{i+1}, g_i^* + g_{i+1}, \Delta_{i+1})$ and delete t_i and all of its descendants.

Proposition A.2 (Proposition 4 in (Greenwald & Khanna, 2001))**.** For any node V, the set of all its descendants in the tree forms a contiguous segment in $S(X)$.

A.5 Formal guarantee

We briefly recall the main guarantees of the GK sketch that we appeal to in the main body of this work and provide a proof for the statements that are not reproduced from previous work.

Lemma 3.8. *[GK guarantees (Greenwald & Khanna, 2001)] Given a data stream $x_1, \ldots, x_n$ of elements drawn from $\mathcal{X}$, the GK sketch outputs a list $S(X)$ of $s(= O((1/\alpha)\log(\alpha n)))$ many tuples $(v_i, g_i, \Delta_i) \in (\mathcal{X} \times \mathbb{N}) \times \mathbb{N} \times \mathbb{N}$ for $i = 1, \ldots, s$ such that if X is the data multi-set then: (1) $\mathrm{rank}(X, \mathrm{val}(v_i)) \subset [\sum_{j \leq i} g_j, \Delta_i + \sum_{j \leq i} g_j]$. (2) $g_i + \Delta_i \leq 2\alpha n$. (3) The first tuple is $(\min\{x \in X\}, 1, 0)$ and the last tuple is $(\max\{x \in X\}, 1, 0)$. (4) The v_i are sorted in ascending order. Without loss of generality, the lower interval bounds $\sum_{j \leq i} g_j$ and upper interval bounds $\Delta_i + \sum_{j \leq i} g_j$ are also sorted in increasing order.*

Proof. The first three statements are part of the GK sketch guarantee. For the third statement, i.e., to see that the v_i are sorted in ascending order, we see that the GK sketch construction ensures that $\mathrm{val}(v_i) \leq \mathrm{val}(v_{i+1})$ for all i. Since an insertion operation always inserts a repeated value after all previous occurrences and the tuple order is always preserved, it follows that $\mathrm{ix}(v_i) \leq \mathrm{ix}(v_{i+1})$ as well, so in sum $\mathrm{rank}(X, v_i) \leq \mathrm{rank}(X, v_{i+1})$. In other words, the sort order in the GK sketch is *stable*.

The fact that the sequence $\sum_{j \leq i} g_j$ is sorted in increasing order follows from the non-negativity of the g_i. To ensure that $\Delta_i + \sum_{j \leq i} g_j$ are sorted in increasing order note that we always have that $\mathrm{rank}(X, v_i) \leq \mathrm{rank}(X, v_{i+1})$ so that we can decrement Δ_i and ensure that $\Delta_i + \sum_{j \leq i} g_j \leq \Delta_{i+1} + \sum_{j \leq i+1} g_j$ without violating the guarantees of the GK sketch. □

B Omitted proofs

We first recall and add some definitions that we will require in the sequel.

Definition B.1. Let $X = ((x_1, 1), \ldots, (x_n, n))$ (sometimes implicitly referred to as $X = (x_1, \ldots, x_n)$) be a stream of elements drawn from some finite totally ordered data universe $\mathcal{X}$, i.e., $x_i \in \mathcal{X}$ for all $i \in [n]$.

1. **Rank**: Given a totally ordered finite data universe $\mathcal{X}$, a data set X and a value $x \in \mathcal{X}$, let $\mathrm{rank}_X(x) = \mathrm{rank}(X, x) = \sum_{y \in X} \mathbb{1}[y \leq x]$.
2. For $(x_i, i) \in X$, let $\mathrm{val}((x_i, i)) = x_i$ and $\mathrm{ix}((x_i, i)) = \sum_{j \leq i} |\{(x_j, j) : x_j < x_i \text{ or } x_j = x_i, j < i\}|$.
3. For $v_1, v_2 \in X$, we say that $v_1 \leq v_2$ if $\mathrm{ix}(v_1) \leq \mathrm{ix}(v_2)$.
4. The q-quantile of X is $\mathrm{val}(v)$ for $v \in X$ with $\mathrm{ix}(v) = \lceil qn \rceil$.
5. For $x \in \mathcal{X}$, we define $r_{\min}(x) = |\{v \in X : \mathrm{val}(v) < x\}|$, $r_{\max}(x) = |\{v \in X : \mathrm{val}(v) \leq x\}|$ and $\mathrm{rank}(X, x)$ to be the interval $[r_{\min}(x), r_{\max}(x)]$.
6. We say that $x \in \mathcal{X}$ is an α-approximate q-quantile for X if $\mathrm{rank}(X, x) \cap [\lceil qn \rceil - \alpha n, \lceil qn \rceil + \alpha n] \neq \emptyset$.

With this notation, the data set X is naturally identified as a multi-set of elements drawn from $\mathcal{X}$.

Definition B.2. Let $\hat{r}_{\min}(x) = \max\{\sum_{j \leq i} g_j : \mathrm{val}(v_i) < x\}$ and $\hat{r}_{\max}(x) = \min\{\Delta_i + \sum_{j \leq i} g_j : \mathrm{val}(v_i) > x\}$. Note that for every $v \in X$ such that $\mathrm{val}(v) = x$, $\mathrm{ix}(v) \in [\hat{r}_{\min}(x), \hat{r}_{max}(x)]$.

B.1 Proofs of privacy and utility for `DPExpGK`

Outline: From Definition B.2 to Lemma B.5, we formalize how the GK sketch may be used to construct rank interval estimates for any data domain value. We then recall and apply the exponential mechanism with a score function derived from the GK sketch (Definition B.6 to Lemma B.9 and Algorithm 1), and derive the error guarantee Lemma B.11. We conclude this subsection with a detailed description of an efficient implementation of the exponential mechanism (Algorithm 8 and Lemma B.12), and summarize our final accuracy and space complexity guarantees in Theorem 4.1.

Remark B.3. We can add two additional tuples $(-\infty, 0, 0)$ and $(\infty, 0, 0)$ to the sketch, which corresponds to respective rank intervals $[0, 0]$ and $[n+1, n+1]$. The bounds $g_i + \Delta_i \leq 2\alpha n$ are preserved. This will ensure that the sets $\{i : \mathrm{val}(v_i) < x\}$ and $\{i : \mathrm{val}(v_i) > x\}$ for any $x \in \mathcal{X}$ are always non-empty.

We formalize the rank interval estimation in a partition-wise manner as below.

Lemma B.4. *Given a GK sketch $(v_1, g_1, \Delta_1), \ldots, (v_s, g_s, \Delta_s)$, for every $x \in \mathcal{X}$ one of the following two cases holds:*

1. *$x = v_i$ for some $i \in [s]$ and $i = \min\{j : v_j = x\}$,*

$$\hat{r}_{\min}(x) = \sum_{j \leq i} g_j$$
$$\hat{r}_{\max}(x) = \min\{\Delta_{i^*} + \sum_{j \leq i^*} g_j : \exists i^*, \mathrm{val}(v_{i^*}) > \mathrm{val}(v_i)\}$$

2. *$x \in (v_{i-1}, v_i)$, i.e., $x > v_{i-1}$ and $x < v_i$ for some $i \in [s]$,*

$$\hat{r}_{\min}(x) = \sum_{j \leq i} g_j$$
$$\hat{r}_{\max}(x) = \Delta_{i+1} + \sum_{j \leq i+1} g_j$$

Proof. Recall, by Remark B.3, that the first tuple and the last tuple are formal elements at $-\infty$ and ∞, ensuring that every data universe element either explicitly occurs in the GK sketch or lies between two values that occur in the GK sketch. Both statements now follow directly from Definition B.2 and the fact that the values v_i occur in increasing order in the sketch (Lemma 3.8). □

We bound the quality of the rank interval estimate $[\hat{r}_{\min}(x), \hat{r}_{\max}(x)]$ compared to the true rank interval $[r_{\min}(x), r_{\max}(x)]$ as follows.

Lemma B.5. *$|r_{\min}(x) - \hat{r}_{\min}(x)| \leq 2\alpha n$ and $|r_{\max}(x) - \hat{r}_{\max}(x)| \leq 2\alpha n$.*

Proof. Let $i^* = \mathrm{argmax}_{i:\mathrm{val}(v_i)<x} \sum_{j \leq i} g_j$. Then by Definition of i^*, we have that $\mathrm{val}(v_{i^*}) < x \leq \mathrm{val}(v_{i^*+1})$. It follows that

$$\begin{aligned}[\mathrm{ix}(v_{i^*}), \mathrm{ix}(v_{i^*+1})] &\subset [\sum_{j \leq i^*} g_j, \Delta_{i^*+1} + g_{i^*+1} + \sum_{j \leq i^*} g_j] \\ &\subset [\hat{r}_{\min}(x), \Delta_{i^*+1} + g_{i^*+1} + \hat{r}_{\min}(x)]\end{aligned}$$

Since $r_{\min}(x) \in [\mathrm{ix}(v_{i^*}), \mathrm{ix}(v_{i^*+1})]$ and $g_{i^*+1} + \Delta_{i^*+1} \leq 2\alpha n$, it follows that $|r_{\min}(x) - \hat{r}_{\min}(x)| \leq 2\alpha n$. The other inequality follows analogously. □

Definition B.6 (Exponential Mechanism (McSherry & Talwar, 2007))**.** Let $u : \mathcal{S}^S \times \mathcal{R} \to \mathbb{R}$ be an arbitrary score function with global sensitivity δ_u. For any database summary $d \in \mathcal{S}^S$ and privacy parameter $\epsilon > 0$, the exponential mechanism $\mathcal{E}_u^\epsilon : \mathcal{S}^S \to \mathcal{R}$ outputs $r \in \mathcal{R}$ with probability $\propto \exp(\frac{\epsilon \cdot u(S(X), r)}{2\delta_u})$ where

$$\delta_u = \max_{X \sim X', r} |u(S(X), r) - u(S(X'), r)|.$$

The following statement formalizes the trade-off between the privacy parameter ϵ and the tightness of the tail bound on the score attained by the exponential mechanism.

Theorem B.7 ((McSherry & Talwar, 2007; Smith, 2011)). *The exponential mechanism (Definition B.6) satisfies ϵ-differential privacy. Further, the following tail bound on the utility (the score of the output element) holds:*

$$P\left(u(S(X), \mathcal{E}_u^\epsilon(S(X))) < \max_{r \in R} u(S(X), r) - \frac{2\delta_u(t + \ln s)}{\epsilon}\right) \leq e^{-t},$$

where s is the size of the universe from which we are sampling from.

To run the exponential mechanism using our approximate rank interval estimates, we define a score function as follows.

Definition B.8. Let $d(\cdot, \cdot)$ denote the ℓ_1 metric on $\mathbb{R}$. Given a sketch $S(X)$, we define a score function on $\mathcal{X}$:

$$\begin{aligned} u(S(X), x) &= -\min\{|y - \lceil qn \rceil| : y \in [\hat{r}_{\min}(x), \hat{r}_{\max}(x)]\} \\ &= -d(\lceil qn \rceil, [\hat{r}_{\min}(x), \hat{r}_{\max}(x)]) \end{aligned}$$

The magnitude of the noise that is added in the course of the exponential mechanism depends on the sensitivity of the score function, which we bound from above as follows.

Lemma B.9. *For all $n > 1/\alpha$, the sensitivity of u (i.e., δ_u) is at most $4\alpha n + 2$ units.*

Proof. Fix any data set X' neighbouring X under swap DP and let $[r'_{\min}(\cdot), r'_{\max}(\cdot)]$ be the rank ranges with respect to X' for values in $\mathcal{X}$. Let $[\hat{r'}_{\min}(\cdot), \hat{r'}_{\max}(\cdot)]$ denote the confidence interval derived from the GK sketch $S(X')$ for values in $\mathcal{X}$.

Claim B.10. $|r_{\min}(x) - r'_{\min}(x)| \leq 2$, $|r_{\max}(x) - r'_{\max}(x)| \leq 2$.

Proof. These bounds follow directly from the Definition of $r_{\min}$ and $r_{\max}$; under swap DP at most two elements of the stream are changed which implies that the count of the sets defining these terms changes by at most 1 unit each for a total shift of 2 units (in fact, this can be bounded by 1 unit). □

We now prove the sensitivity bound.

$$\begin{aligned} u(S(X), x) &= -d(\lceil qn \rceil, [\hat{r}_{\min}(x), \hat{r}_{\max}(x)]) \\ &\leq -d(\lceil qn \rceil, [r_{\min}(x), r_{\max}(x)]) + 2\alpha n \\ &\leq -d(\lceil qn \rceil, [r'_{\min}(x), r'_{\max}(x)]) + 2\alpha n + 2 \\ &\leq -d(\lceil qn \rceil, [\hat{r'}_{\min}(x), \hat{r'}_{\max}(x)]) + 4\alpha n + 2 \\ &\leq u(S(X'), x) + 4\alpha n + 2. \end{aligned}$$

Swapping the positions of X and X', we get the reverse bound to complete the sensitivity analysis. □

We can now derive a high probability bound on the utility that is achieved by Algorithm 1.

Lemma B.11. *If $\hat{x}$ is the value returned* `DPExpGK` *then with probability $1 - \beta$,*

$$d(\lceil qn \rceil, [\hat{r}_{\min}(\hat{x}), \hat{r}_{\max}(\hat{x})]) \leq 2\alpha n + \frac{2(4\alpha n + 2)\log(|\mathcal{X}|/\beta)}{\epsilon}.$$

Proof. By construction, Algorithm 1 is simply a call to the exponential mechanism with score function $u(S(X), \cdot)$, Since for any target q-quantile, $\lceil qn \rceil$ lies in $[0, n]$ it follows that there is some $i^* \in s$ such that $\lceil qn \rceil \in [\sum_{j \leq i^*} g_j, \Delta_{i^*+1} + g_{i^*+1} + \sum_{j \leq i^*} g_j]$. It follows that $d(\lceil qn \rceil, [r_{\min}(\text{val}(v_{i^*}), r_{\max}(\text{val}(v_{i^*}))]) \leq 2\alpha n$

and that hence $\max(u(S(X), x)) \geq -2\alpha n$. If x^* is the output of the exponential mechanism, then applying the utility tail bound we get that with probability $1 - \beta$,

$$u(S(X), x^*) \geq -2\alpha n - \frac{2(4\alpha n + 2)\log(|\mathcal{X}|/\beta)}{\epsilon}.$$

By definition of u, the desired bound follows. □

Algorithm 8: `DPExpGKGumb`: Implementing the Exponential Mechanism on $S(X)$ using the Gumbel Distribution (Optimized implementation of algorithm 1)

Data: $X = (x_1, x_2, \ldots, x_n)$
Input: ϵ, α (approximation parameter), $q \in [0, 1]$ (quantile parameters)

Build summary sketch $S(X)$ and let $s = |S(X)|$.
Let $(v_i, g_i, \Delta_i) = S(X)[i]$ for all $i \in [s]$
maxIndex $= -1$
maxValue $= -\infty$
`/* Iterating over tuple values` v_i `*/`
Let $i = 1$
while $i <= s$ **do**
  $\hat{r}_{\min} = \sum_{j \leq i} g_j$
  $\hat{r}_{\max} = \min\{\Delta_{i^*} + \sum_{j \leq i^*} g_j : \mathrm{val}(v_{i^*}) > \mathrm{val}(v_i)\}$
  $u_i = -\min\{|y - \lceil qn \rceil| : y \in [\hat{r}_{\min}, \hat{r}_{\max}]\}$.
  $f = \frac{\epsilon}{2} u_i$
  $\tilde{f} = f + \mathrm{Gumb}(0, 1)$
  if $\tilde{f} >$ *maxValue* **then**
    maxIndex $= (i, \text{tuple})$
    maxValue $= \tilde{f}$
  $i \leftarrow \min\{j : v_j > v_i\}$
`/* Iterating over intervals between tuples` $\mathcal{X}(v_{i-1}, v_i) \subset \mathcal{X}$ `*/`
Let $i = 1$
while $i <= s$ **do**
  if $\mathcal{X}(v_{i-1}, v_i)$ *is not empty* **then**
    $\hat{r}_{\min} = \sum_{j \leq i} g_j$
    $\hat{r}_{\max} = \Delta_{i+1} + \sum_{j \leq i+1} g_j$
    $u_{i-1,i} = -\min\{|y - \lceil qn \rceil| : y \in [\hat{r}_{\min}, \hat{r}_{\max}]\}$.
    $f = \log(|\mathcal{X}(v_{i-1}, v_i)|) + \frac{\epsilon}{2} u_{i-1,i}$
    $\tilde{f} = f + \mathrm{Gumb}(0, 1)$
    if $\tilde{f} >$ *maxValue* **then**
      maxIndex $= (i, \text{interval})$
      maxValue $= \tilde{f}$
  $i \leftarrow i + 1$
if *maxIndex* $= (i, \textit{tuple})$ *for some* $i \in 1, \ldots, s$ **then**
  return v_i
else if *maxIndex* $= (i, \textit{interval})$ *for some* $i \in 1, \ldots, s$ **then**
  Pick $v \in \mathcal{X}(v_{i-1}, v_i)$ uniformly at random
  return v

As discussed before, in general a naive implementation of the exponential mechanism as in Algorithm 1 would in general not be efficient. To resolve this issue, in Algorithm 8 we take advantage of the partition of the data domain by the score function and the *Gumbel-max* trick to implement the exponential mechanism without

any higher-order overhead and return an α-approximate q quantile. This trick has become a standard way to implement the exponential mechanism over intervals/tuples.

Lemma B.12. *Algorithm 8 implements algorithm 1 on the data universe $\mathcal{X}$ with space complexity $O(|S(X)|)$ (where $S(X)$ is the GK sketch) and additional time complexity $O(|S(X)| \log |S(X)|)$.*

Proof. We see that it will suffice to show that algorithm 8 executes the exponential mechanism with the same score function as algorithm 1 to prove that it is a valid implementation of the latter.

As noted in previous work (Abernethy et al., 2017), if $Z_1, \ldots, Z_N$ are drawn i.i.d. from standard Gumbel distribution, then

$$\mathbb{P}\left[f_i + Z_i = \max_{j \in [N]}\{f_j + Z_j\}\right] = \frac{\exp(f_i)}{\sum_{j \in [N]} \exp(f_j)}, \forall i \in [N].$$

We recall that when running the exponential mechanism on $\mathcal{X}$, we want to sample the element $x \in \mathcal{X}$ with probability $\propto \exp(\epsilon u(S(X), x)$. To implement the exponential mechanism via the identification with Gumbel argmax distribution above, we will simply compute the scores $u(S(X), x)$ and let $f_i = \epsilon \cdot u(S(X), x)$.

For $x \in \mathcal{X}$ such that $x = v_i$ for some $i \in [s]$, Algorithm 8 directly computes the scores according to Definition B.8 and Lemma B.4; this is formalized by lines 5 to 15 in the pseudo code.

For $x \in \mathcal{X}$ which lie strictly between the tuple values $\{v_i : i \in [s]\}$, we proceed as follows. Fixing i, from Lemma B.4 we have that that for $\mathcal{X}(v_{i-1}, v_i) := \{x \in \mathcal{X} : x > v_{i-1}, x < v_i\}$, the rank confidence interval estimate is the same, i.e. $[\sum_{j \leq i-1} g_j, \Delta_i + \sum_{j \leq i} g_j]$. It follows from Definition B.8 that for all such domain values the the score function value $u(S(X), \cdot)$ is equal; this is denoted $u_{i-1,i}$ in the pseudo code. By summing the probabilities for sampling individual domain elements, it follows that the likelihood of the exponential mechanism outputting some value from the set $\mathcal{X}(v_{i-1}, v_i)$ is $\propto |\mathcal{X}(v_{i-1}, v_i)| \exp(\epsilon u_{i-1,i}/2) = \exp(\epsilon u_{i-1,i}/2 + \log(|\mathcal{X}(v_{i-1}, v_i)|))$. This is formalized by lines 16 to 27 in the pseudo code.

Finally, if some interval is selected, then by outputting elements chosen uniformly at random, we ensure that the likelihood of $x \in \mathcal{X}(v_{i-1}, v_i)$ being output is $\propto \frac{1}{|\mathcal{X}(v_{i-1}, v_i)|} \cdot \exp(\epsilon u_{i-1,i}/2 + \log(|\mathcal{X}(v_{i-1}, v_i)|) = \exp(\epsilon u_{i-1,i})$. Note that we do not need to account for ties in the Gumbel scores as the event $f_i + Z_i = f_j + Z_j$ for any $j \neq i$ has measure 0. [5]

To bound the space and time complexity; we note that by the guarantees of the GK sketch, the size of the sketch $S(X)$ is $O((1/\alpha) \log \alpha n)$; we compute Gumbel scores by iterating over tuples and intervals of which there are at most $O(S(X))$-many of each, each computation takes at most $O(\log |S(X)|)$ time, and only the max score and index seen at any point is tracked in the course of the algorithm. □

We can now state and prove our main theorem in this section, proving utility bounds for α-approximating quantiles through `DPExpGK` with sublinear space.

Theorem 4.1. *Algorithm 1 is ϵ-differentially private. Let $\hat{x}$ be the value returned by Algorithm 1 when initialized with target quantile q. The following statements hold:*

1. *Algorithm 1 can be implemented to run with space complexity $O((1/\alpha) \log \alpha n)$, such that with probability $1 - \beta$*

$$d(\lceil qn \rceil, [\hat{r}_{\min}(\hat{x}), \hat{r}_{\max}(\hat{x})]) \leq 2\alpha n + \frac{2(4\alpha n + 2) \log(|\mathcal{X}|/\beta)}{\epsilon} \tag{2}$$

2. *For $n > \frac{\log |\mathcal{X}|/\beta}{12\alpha\epsilon}$, Algorithm 1 can be implemented to run with space complexity $O\left((\alpha\epsilon)^{-1} \log(|\mathcal{X}|/\beta) \log(\alpha\epsilon n)\right)$ such that with probability $1 - \beta$, $\hat{x}$ is an α approximate q-quantile.*

[5] As is usual in the privacy literature, we assume that the sampling of the $Gumb(0, 1)$ distribution can be done on finite-precision computers (Balcer & Vadhan, 2018). While the problem of formally dealing with rounding has not been settled in the privacy literature (Mironov, 2012), for any practical purpose it easily suffices to store the output of the Gumbel distribution using a few computer words.

Our dependence in α, which for practical purposes is usually the most important term, is optimal. Recent subsequent work by Kaplan and Stemmer (Kaplan & Stemmer, 2021) shows how to improve the dependence in other parameters if approximate (rather than pure) differential privacy is allowed, or if the stream length is large enough.

Proof. The privacy guarantee of Algorithm 1 follows from the privacy guarantee of the exponential mechanism and Lemma B.12. The accuracy bound in equation 2 is simply a restatement of Lemma B.11, and the space complexity bounds follow from the GK sketch space complexity bound. To derive the second statement, we substitute $\frac{\alpha \min\{\epsilon,1\}}{24\log(|\mathcal{X}|/\beta)}$ for the approximation parameter α in equation 2 and get

$$\begin{aligned}
&d(\lceil qn \rceil, [\hat{r}_{\min}(\hat{x}), \hat{r}_{\max}(\hat{x})]) \\
&\leq 2 \cdot \frac{\alpha \min\{\epsilon,1\} n}{24\log(|\mathcal{X}|/\beta)} + \frac{2(4(\frac{\alpha \min\{\epsilon,1\}}{24\log(|\mathcal{X}|/\beta)})n + 2)\log(|\mathcal{X}|/\beta)}{\epsilon} \\
&\leq \frac{\alpha n}{12\log(|\mathcal{X}|/\beta)} + \frac{\alpha n}{3} + \frac{4\log|\mathcal{X}|/\beta}{\epsilon} \\
&\leq \frac{\alpha n}{12} + \frac{\alpha n}{3} + \frac{\alpha n}{3} \\
&\leq \alpha n.
\end{aligned}$$

The space complexity bound now follows directly from the space complexity bound derived in Lemma B.12, the space complexity bound $O((1/\alpha)\log \alpha n)$ for the GK sketch, and by substituting $\frac{\alpha \min\{\epsilon,1\}}{24\log(|\mathcal{X}|/\beta)}$ for α. □

B.2 Proofs of privacy and utility for `DPHistGK`

In this section we prove theorem 4.2, which we restate here for ease of reference.

Theorem 4.2. *For the one-dimensional normal distribution $\mathcal{N}(\mu, \sigma^2)$, let $S = (X_1, X_2, \ldots, X_n)$ be a data stream through which we wish to obtain $\tilde{Q}_S^q$, a DP estimate of the q-quantile of the distribution.*

For any $q \in (0,1)$, there exists an (ϵ, δ)-DP algorithm such that, with probability at least $1-\beta$, we obtain $|Q_{\mathcal{D}}^q - \tilde{Q}_S^q| \leq \alpha$ for any $\alpha > 0$, $\beta \in (0,1], \epsilon, \delta \in (0, 1/n)$ and for stream length

$$n \geq \max\left\{\min\left\{A, B\right\}, C\right\}, \quad \textit{where}$$

$$A = O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{R}{\sigma\beta}\right), B = O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{1}{\beta\delta}\right), C = O\left(\frac{R^2}{\sigma^2\alpha^2}\log\frac{1}{\beta}\right)$$

as long as $\mu \in (-R, R)$ and using space of $O(\max\{\frac{R}{\sigma}, \frac{1}{\alpha}\log \alpha n\})$.

Proof. For any stream $S = (X_1, \ldots, X_n)$, we use the triangle inequality so that

$$\begin{aligned}
|Q_{\mathcal{D}}^q - \tilde{Q}_S^q| &\leq |Q_{\mathcal{D}}^q - Q_S^q| + |Q_S^q - \tilde{Q}_S^q| && (3)\\
&\leq \alpha/2 + \alpha/2. && (4)
\end{aligned}$$

$|Q_{\mathcal{D}}^q - Q_S^q| \leq \alpha/2$ follows with probability $1-\beta/2$ by Corollary B.14 and $|Q_S^q - \tilde{Q}_S^q| \leq \alpha/2$ follows with probability $1-\beta/2$ by Lemma B.15. The space complexity follows with probability 1 via the deterministic nature of the Greenwald-Khanna sketch. □

Lemma B.13 (Dvoretzky-Kiefer-Wolfowitz inequality (Dvoretzky et al., 1956))**.** *For any $n \in \mathbb{Z}_+$, let $X_1, \ldots, X_n$ be i.i.d. random variables with cumulative distribution function F so that $F(x)$ is the probability that a single random variable X is less than x for any $x \in \mathbb{R}$. Let the corresponding empirical distribution function be $F_n(x) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}[X_i \leq x]$ for any $x \in \mathbb{R}$. Then for any $\gamma > 0$,*

$$\mathbb{P}\left(\sup_{x\in\mathbb{R}} |F_n(x) - F(x)| > \gamma\right) \leq 2\exp(-2n\gamma^2).$$

Corollary B.14. *For any $q \in (0,1)$, let $Q^q_{\mathcal{D}}$ be the q-quantile estimate for the distribution $\mathcal{D}$ and Q^q_S be the q-quantile estimate for the sample. Then, $|Q^q_{\mathcal{D}} - Q^q_S| \leq \alpha/2$ with probability $1-\beta/2$ when $n \geq \frac{2}{\alpha^2}\log 4/\beta$.*

Proof. Follows by the DKW inequality (Lemma B.13) where $n \geq \frac{1}{2\gamma^2}\log 4/\beta$ and $\gamma = \alpha/2$. □

Lemma B.15. *For any $q \in (0,1)$, $\alpha > 0$, $\beta \in (0,1]$, $\epsilon, \delta \in (0, 1/n)$, there exists an (ϵ,δ)-differentially private algorithm $\tilde{Q}^q_S$ for computing the q-quantile such that*

$$|Q^q_S - \tilde{Q}^q_S| \leq \alpha/2,$$

with probability $\geq 1-\beta$ for stream length

$$n \geq O\left(\min\{O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{R}{\sigma\beta}\right), O\left(\frac{R}{\epsilon\sigma\alpha}\log\frac{1}{\beta\delta}\right)\}\right).$$

Furthermore, with probability 1, $\tilde{Q}^q_S$ uses space of $O(\max\{\frac{R}{\sigma}, \frac{1}{\alpha}\log\alpha n\})$.

Proof. First, by the tail bounds of the Gaussian distribution (Claim B.17), we can obtain that for any $i \in [n]$,

$$\mathbb{P}[|X_i - \mu| > c] \leq 2e^{-c^2/2\sigma^2},$$

so that by the union bound,

$$\mathbb{P}[\exists i, |X_i - \mu| \geq c] \leq 2ne^{-c^2/2\sigma^2},$$

which implies that for any $\beta \in (0,1]$,

$$\mathbb{P}[\forall i, |X_i - \mu| \leq \sigma\sqrt{2\log 4n/\beta}] \geq 1-\beta/2,$$

which holds by our sample complexity (stream length) guarantees.

Next, let $r = \lceil R/\sigma \rceil$. [6] Divide $[-R-\sigma/2, R+\sigma/2]$ into $2r+1$ bins of length at most σ each. Each bin B_j should equal $((j-0.5)\sigma, (j+0.5)\sigma]$ for any $j \in \{-r, \ldots, r\}$. Next run the histogram learner of Lemma B.16 with per-bin accuracy parameter of α/K, high-probability parameter of $\beta/2$, privacy parameters $\epsilon, \delta \in (0, 1/n)$, and number of bins $K = 2\lceil R/\sigma \rceil + 1$. We can do this because of our sample complexity (stream length) bounds. Then we obtain noisy estimates $\tilde{p}_{-r}, \ldots, \tilde{p}_r$ with per-bin accuracy of α/K. Then any quantile estimate would have accuracy of α (by summing noisy estimates for at most K bins).

Next, we use these bins to construct a sketch (private by DP post-processing) based on the deterministic algorithms of (Greenwald & Khanna, 2004) to, with probability 1, obtain space of $O(\max\{\frac{R}{\sigma}, \frac{1}{\alpha}\log\alpha n\})$. □

Lemma B.16 (Histogram Learner (Bun et al., 2015; Vadhan, 2017; Karwa & Vadhan, 2018))**.** *For every $K \in \mathbb{N} \cup \{\infty\}$ and every collection of disjoint bins $B_1, \ldots, B_K$ defined on the domain $\mathcal{X}$. For any $n \in \mathbb{N}$, $\epsilon, \delta \in (0, 1/n), \alpha > 0$, and $\beta \in (0,1)$, there exists an (ϵ,δ)-DP algorithm $M : \mathcal{X}^n \to \mathbb{R}^K$ such that for every distribution $\mathbb{D}$ on the domain $\mathcal{X}$, if*

1. *$X_1, \ldots, X_n \sim \mathbb{D}$, $p_k = \mathbb{P}[X_i \in B_k]$ for any $k \in [K]$,*
2. *$(\tilde{p}_1, \ldots, \tilde{p}_K) \leftarrow M(X_1, \ldots, X_n)$,*
3. *$n \geq \max\left\{\min\left\{\frac{8}{\epsilon\alpha}\log\frac{2K}{\beta}, \frac{8}{\epsilon\alpha}\log\frac{4}{\beta\delta}\right\}, \frac{1}{2\alpha^2}\log\frac{4}{\beta}\right\}$,*

then (over the randomness of the data $X_1, \ldots, X_n$ and of M)

1. *$\mathbb{P}_{\underline{X}\sim\mathbb{D},M}[|\tilde{p}_k - p_k| \leq \alpha] \geq 1-\beta$,*
2. *$\mathbb{P}[\text{argmax}_k \tilde{p}_k = j] \leq np_j$ if $K \geq 2/\delta$,*

[6] Note that this argument is similar to the arguments for Algorithm 1 in (Karwa & Vadhan, 2018).

3. $\mathbb{P}[\text{argmax}_k \tilde{p}_k = j] \leq np_j + 2\exp(-(\epsilon n/8) \cdot (\max_k p_k))$ *if* $K < 2/\delta$.

Claim B.17 (Gaussian Tail Bound)**.** *Let Z be a random variable distributed according to a standard normal distribution (with mean 0 and variance 1). For every $t > 0$,*

$$\mathbb{P}[|Z| > t] \leq 2\exp(-t^2/2).$$

Lemma B.18. *Algorithm 2 satisfies $(\epsilon, 0)$-DP.*

Proof. For any $i \in [n]$, any item x_i can belong in at most one bin. Plus, the global sensitivity of the function that computes the empirical histogram is 2, since changing a single item can change the contents of at most two bins.

As a result, adding noise of $\text{Lap}(0, 2/\epsilon)$ to each bin satisfies ϵ-DP by Theorem B.19. □

Theorem B.19 (Laplace Mechanism (Dwork et al., 2006))**.** *Fix $\epsilon > 0$ and any function $f : \mathcal{Y}^n \to \mathbb{R}^K$. The Laplace mechanism outputs*

$$f(y) + (L_1, \ldots, L_K),$$

$L_1, \ldots, L_K \sim \text{Lap}(0, GS_f/\epsilon)$ where GS_f is the global sensitivity of the function f. Furthermore, the mechanism satisfies $(\epsilon, 0)$-DP.

B.3 Extension to the Continual Observation setting

In this section we prove theorem 5.1, which formalizes the privacy and utility guarantees of algorithm 3 in the continual observation setting. We start by making some definitions that will aid us in our analysis.

Definition B.20. We make the following definitions:

1. **Stream Prefix**: Given an input data stream $X = (x_1, \ldots, x_n)$ we define the prefix up to the index s element of this stream $X[1 : s] := (x_1, \ldots, x_s)$. Note that we can overload notation and treat $X[1 : s]$ as a data set by ignoring the order in which the elements arrive.
2. **Checkpoint**: A set $S \neq \emptyset$ is a set of checkpoints for stream X if $\forall j \in S$, $\exists$ value v_j that is a $(\alpha/2)$-approximate q-quantile for $X[1 : j]$.

We first observe that if we have an $\alpha/2$ approximate quantile for a given data set, then that estimate remains at least an α-approximate quantile for a slightly larger set as well.

Lemma B.21. *If $x \in X$ is an $\alpha/2$-approximate q-quantile for a data set X (for some $q \in [0, 1]$), then it is an α-approximate q-quantile for any data set $X' \supset X$ such that $|X'| \leq (1 + \alpha/2)|X|$.*

Proof. Since x is an $\alpha/2$-approximate q-quantile, we have that

$$(1 - \alpha/2)|X| \leq \underset{X}{\text{rank}}(x) \leq (1 + \alpha/2)|X|.$$

We then have that

$$\begin{aligned}
\underset{X'}{\text{rank}}(x) &= \sum_{y \in X'} \mathbb{1}[y \leq x] \\
&= \sum_{y \in X} \mathbb{1}[y \leq x] + \sum_{y \in X' \setminus X} \mathbb{1}[y \leq x] \\
\Rightarrow \sum_{y \in X} \mathbb{1}[y \leq x] &\leq \sum_{y \in X'} \mathbb{1}[y \leq x] \leq \sum_{y \in X} \mathbb{1}[y \leq x] + \sum_{y \in X' \setminus X} \mathbb{1}[y \leq x] \\
\Rightarrow (1 - \alpha/2)|X| &\leq \sum_{y \in X'} \mathbb{1}[y \leq x] \leq (1 + \alpha/2)|X| + (|X'| - |X|)
\end{aligned}$$

$$\Rightarrow (1-\alpha/2)|X| \leq \sum_{y \in X'} \mathbb{1}[y \leq x] \leq (1+\alpha/2)|X| + \alpha|X|/2$$
$$\Rightarrow (1-\alpha)|X| \leq \sum_{y \in X'} \mathbb{1}[y \leq x] \leq (1+\alpha)|X|,$$

i.e., x is also an α-approximate q-quantile for X', as required. □

We now show that the choice of checkpoints made in the pseudocode of algorithm 3 is valid as per our definition.

Lemma B.22. *In algorithm 3, the set $\{\mathrm{cp}(s') : s' \in [n]\}$ forms a valid set of checkpoints for the data stream X. More concretely v_s is an $\alpha/2$-approximate q-quantile for $X[1:s]$.*

Proof. First we bound the size of the set of checkpoints $\{\mathrm{cp}(s') : s' \in [n]\}$. Since a new checkpoint value is generated only when $s \geq (1+\alpha/2)\mathrm{cp}(s-1)$, it follows that for any new checkpoint where $\mathrm{cp}(s') \neq \mathrm{cp}(s'-1)$, we have $\mathrm{cp}(s') \geq (1+\alpha/2)\mathrm{cp}(s'-1)$. Using Theorem 4.1, we set the first checkpoint value to $\frac{\log|\mathcal{X}|/\beta^*}{12\alpha^*\epsilon^*}$, where α^*, β^* and ϵ^* are the accuracy parameter, the failure probability, and the private parameter that are passed in the calls to `DPExpGK`, respectively. [7] It follows that there are at most $k \leq \log_{1+\alpha/2} n = \frac{3}{\alpha}\log n$ checkpoints using the fact that for all $x > -1$, $\frac{x}{1+x} \leq \log(1+x) \leq x$.

Since $v_{\mathrm{cp}(s)}$ is the output of `DPExpGK` given a GK sketch with accuracy parameter α^* and privacy parameter ϵ^* it follows that with probability $1-\beta^*$ where $\beta^* = O(\alpha\beta/\log n)$, $\mathrm{rank}_{X[1:s']} \in (1-\alpha/2, 1+\alpha/2)qn$. We now apply the union bound over all private approximate quantile computations at checkpoints and are done. □

We can now prove our main technical result.

Theorem 5.1. *Let $\epsilon, \alpha > 0$, $n \in \mathbb{Z}$. For any $\beta \in (0,1]$, with probability $\geq 1-\beta$, Algorithm 3 maintains an α-approximate q-quantile at every point s in the data stream such that $s = \Omega\left(\frac{1}{\epsilon\alpha^2}\log^2 n \log\left(\frac{|\mathcal{X}|\log n}{\alpha\beta}\right)\right)$. Furthermore, Algorithm 3 satisfies ϵ-DP.*

Proof. To see that Algorithm 3 is ϵ-DP, we observe that the output of this algorithm throughout the data stream can be summarized by its outputs at the checkpoints $\{\mathrm{cp}(s) : s \in [n]\}$ (the points in the stream at which a new checkpoint is reached and a new value released are known publicly, so this suffices for privacy analysis). It follows that there is a choice of $\epsilon^* = \Omega(\alpha\epsilon/\log n)$ that gives us an ϵ-DP mechanism.

We now prove the accuracy guarantee. For any arbitrary $s \geq \frac{\log|\mathcal{X}|/\beta}{12\alpha^*\epsilon^*}$, i.e., the default checkpoint value, the output v_s of the algorithm will equal the value at the most recent checkpoint, i.e., $v_{\mathrm{cp}(s)}$. Then, since $|X[1:s]| = s \leq (1+\alpha/2)\mathrm{cp}(s) \leq (1+\alpha/2)|X[1:\mathrm{cp}(s)]|$, by Lemma B.21 it follows that $\mathrm{rank}_{X[1:s]}(v_s) \in (1-\alpha, 1+\alpha)qn$, i.e., v_s is an α-approximate q-quantile for $X[1:s]$. □

C Additional Experiments

In this section, we include some additional experimental details and results.

Varying ϵ for `DPExpGKGumb`: In Table 1, we vary ϵ and compare `DPExpFull` to `DPExpGKGumb` in terms of average absolute error and execution time (in seconds). We see that `DPExpGKGumb` is significantly faster than `DPExpFull` in terms of average execution time. However, `DPExpGKGumb` incurs larger error because of the approximation factor of $\alpha = 0.0001$.

Histogram-Based Algorithm: We also implemented the `DPHistGK` and tested by varying the bin width of the histogram. In general, we observe that `DPExpGK` has superior performance in terms of minimizing the error for any single quantile query. For example, we obtain averge absolute errors of at least 2.56 for 2 bins and 3.001 for 10 bins for $q = 0.3$ for `DPHistGK`. This suggests that we can rely on `DPExpGK` and its variants for

[7] The last checkpoint might occur at $(1+\alpha/2)n$. We may ignore checkpoints past n.

ϵ	DPExpFull (Error)	DPExpGKGumb (Error)	DPExpFull (Time)	DPExpGKGumb (Time)
0.1	0.0019796	0.002172	0.1330166	0.028163
0.5	0.0004581	0.0013467	0.13113595	0.02804
1	0.000249894	0.00127795	0.1335331	0.0284041
5	0.00007110	0.00151	0.1341164231	0.028285

Table 1: $\alpha = 0.0001$, $n = 10^5$, U(0, 10). Time in Seconds.

general-purpose implementations with minimal error. However, it is possible that certain parameter settings (e.g., bin width) for DPHistGK might yield better performance.

Continual Observation Algorithm: We also implement the continual observation algorithm that builds directly on DPExpGK. We fixed the first stream checkpoint to 10000. For $\alpha = 0.001$ and stream length of $n = 100000$ from $U(0, 10)$, we observe average absolute error of 0.00723 for $\epsilon \in \{0.1, 0.5, 1, 5\}$ over 100 trials. For $\alpha = 0.001$ and stream length of $n = 100000$ from $\mathcal{N}(5, 1)$, we observe average absolute error of 0.00134. However, it is possible that the first few checkpoints might be an important parameter for the error of the sketch in the continual observation setting. We leave the full exploration of this question to future work.

C.1 Full Space Quantile Computation

Without the bounded space requirement (i.e., space sublinear in the stream length), we can use the exponential mechanism with a score function that uses the entire stream of values X. In that case, the sensitivity of the score function is at most 1. We use this as one of the baselines for our experimental validation.

Lemma C.1. *Given any insertion only stream*

$$X = (x_1, x_2, \ldots, x_{n-1}, x_n),$$

the sensitivity of the score function u (under swap differential privacy) is at most 1. i.e., $\delta_u \leq 1$. The function u is defined as $u(X, e) = -|\operatorname{rank}(X, e) - r|$ where r is the approximate $\lfloor q \cdot n \rfloor$ rank of the sketch and $\operatorname{rank}(X, e)$ is the rank of e amongst all values in the stream X.

Proof. Let $|X| = n$. The score function becomes $-|\operatorname{rank}(X, e) - n_q|$ where $n_q = \lfloor q \cdot n \rfloor$. Consider two streams with only one element changed: X, X', denoting the element by x_d. Then at time $d \leq n$, in the second stream x'_d is inserted instead of x_d. In both cases, n_q changes by at most q (in the case of add-remove DP) and for swap DP, n_q remains the same. And for any e, $\operatorname{rank}(X', e)$ would differ from $\operatorname{rank}(X, e)$ by at most 1 since the rank of any element can change by at most 1 after adding, deleting, or replacing an item in the stream. Furthermore, for any $n \geq d$, the rank of any e will differ in X, X' by at most 1 replacing x_d with x'_d can displace the rank of any element by at most 1. Also, the term n_q will remain the same. (Note that in the add-remove privacy definition $n_q = \lfloor q \cdot n \rfloor$ would change to either $\lfloor q \cdot (n+1) \rfloor$ or $\lfloor q \cdot (n-1) \rfloor$.)

The "reverse triangle inequality" says that for any real numbers x and y, $|x - y| \geq ||x| - |y||$. As a result, $-|\operatorname{rank}(X, e) - n_q| + |\operatorname{rank}(X', e) - n_q| \leq |\operatorname{rank}(X', e) - \operatorname{rank}(X, e)| \leq 1$ for any e. □