

Individual Sensitivity Preprocessing for Data Privacy

Rachel Cummings*
Georgia Tech

David Durfee†
Georgia Tech

January 24, 2019

Abstract

The sensitivity metric in differential privacy, which is informally defined as the largest marginal change in output between neighboring databases, is of substantial significance in determining the accuracy of private data analyses. Techniques for improving accuracy when the average sensitivity is much smaller than the worst-case sensitivity have been developed within the differential privacy literature, including tools such as smooth sensitivity, Sample-and-Aggregate, Propose-Test-Release, and Lipschitz extensions.

In this work, we provide a new and general Sensitivity-Preprocessing framework for reducing sensitivity, where efficient application gives state-of-the-art accuracy for privately outputting the important statistical metrics median and mean when no underlying assumptions are made about the database. In particular, our framework compares favorably to smooth sensitivity for privately outputting median, in terms of both running time and accuracy. Furthermore, because our framework is a preprocessing step, it can also be complementary to smooth sensitivity and any other private mechanism, where applying both can achieve further gains in accuracy.

We additionally introduce a new notion of *individual sensitivity* and show that it is an important metric in the variant definition of *personalized differential privacy*. We show that our algorithm can extend to this context and serve as a useful tool for this variant definition and its applications in markets for privacy.

Given the effectiveness of our framework in these important statistical metrics, we further investigate its properties and show that: (1) Our construction is conducive to efficient implementation with strong accuracy guarantees, evidenced by an $O(n)$ implementation for median (with presorted data), and $O(n^2)$ implementation for more complicated functions such as mean, α -trimmed mean, and variance. (2) Our construction is both NP-hard and also optimal in the general setting (3) Our construction can be extended to higher dimensions, although it incurs accuracy loss that is linear in the dimension.

*School of Industrial and Systems Engineering, Georgia Institute of Technology. Email: rachelc@gatech.edu. Supported in part by a Mozilla Research Grant.

†School of Computer Science, Georgia Institute of Technology. Email: ddurfee@gatech.edu. This material is based upon work supported by the National Science Foundation under Grants No. 1718533.

1 Introduction

Differentially private algorithms for data analysis guarantee that any individual entry in a database has only a bounded effect on the outcome of the analysis [DMNS06]. These algorithms ensure that the outcomes on any pair of neighboring databases—that differ in a single entry—are nearly indistinguishable. This is typically achieved by perturbing the analysis or its output, using noise that scales with the magnitude of change in the analysis between neighboring databases. This perturbation necessarily leads to decreased accuracy of the analysis. A fundamental challenge in differentially private algorithm design is to simultaneously satisfy privacy guarantees and provide accurate analysis of the database. Privacy alone can be achieved by outputting pure noise, but this fails to yield useful insights about the data. Intuitively, stronger privacy guarantees should yield weaker accuracy guarantees. Quantifying this privacy-accuracy tradeoff has been one major contribution of the existing differential privacy literature. In the last several years, accurate and differentially private algorithms have been designed for a diverse collection of data analysis tasks (see [DR14] for a survey), and have been implemented in practice by major organizations such as Apple, Google, Uber, and the U.S. Census Bureau. The formal guarantees of differential privacy give sharp contrast to ad hoc privacy measures such as anonymization and aggregation, which have both led to infamous privacy violations [NS08, HSR⁺08].

We formalize data analysis tasks as functions that map from the space of all databases to real-valued outputs. The *global sensitivity* of a function is the worst-case difference in the function’s value between all pairs of neighboring databases. Since differential privacy guarantees must hold for all pairs of neighboring databases, this is the scale of noise that must be added to preserve privacy. Strong bounds on global sensitivity imply that the function is well-behaved over the entire data universe, and often allows for privacy-preserving output with strong accuracy guarantees. However, this worst-case measure allows a single outlier database to significantly skew the accuracy of the privacy-preserving algorithm for all databases. Although it is necessary to preserve the privacy of outlying databases, we would prefer to add less noise for improved accuracy guarantees when the average-case sensitivity is far smaller than the worst-case. A variety of well-known techniques have been employed to address this problem including smooth sensitivity and Sample-and-Aggregate [NRS07], Propose-Test-Release [DL09], and Lipschitz extensions [KNRS13, BBDS13, RS16].

Initial work in this space considered a database-specific definition of sensitivity, known as *local sensitivity*, which is the maximum change in the function’s value between a given database and its neighbors [NRS07, DL09]. Ideally, we would like to add noise that scales with the local sensitivity of each database. This would allow us to add less noise to well-behaved regions of the database universe, and only the outliers would require substantial noise. Unfortunately this procedure does not satisfy differential privacy because the amount of noise added to a given database may be highly disclosive. To avoid this information leakage, [NRS07] defined an intermediate notion of *smooth sensitivity*, which smoothed the amount of noise added across databases to preserve differential privacy once again. This technique was also combined with random subset sampling to give an efficient and private procedure, Sample-and-Aggregate, with strong error guarantees when each database was well-approximated by a random subset of its entries [NRS07].

Later work considered partitioning the database universe into well-behaved and outlying databases. Propose-Test-Release defined this partition with respect to local sensitivity and gave accurate outputs only on databases that were sufficiently far from outliers [DL09]. Propose-Test-Release avoided some of the information leakage issues by outputting NULL for any outlying database, and gave efficient implementations for a variety of important functions. The Lipschitz extension framework instead partitioned according to global sensitivity, by identifying a subset of the data universe where the given function had small global sensitivity. On this subset, the function

of interest is simply a Lipschitz function with the constant defined as the small global sensitivity [KNRS13, BBDS13, RS16]. Extending the Lipschitz function to the remaining data universe achieves a function with small global sensitivity that is identical to the original function on the well-behaved databases. Applying any differential privacy algorithm to this Lipschitz function will allow for the use of a much smaller global sensitivity input and will achieve high accuracy on the well-behaved databases.

In this work, we introduce a Sensitivity-Preprocessing framework that will similarly approximate a given function with a sensitivity bounded function, which we call the Sensitivity-Preprocessing Function. Our Sensitivity-Preprocessing Function will take advantage of the specific metric space structure of the data universe to give a more constructive approach. At a high-level, while Lipschitz extensions are initialized with a well-behaved subset of the data universe, our algorithm will find this well-behaved subset as it constructs the Sensitivity-Preprocessing Function. As a result, our procedure will be much more localized and can always give an exponential-time construction even in the most general setting, whereas Lipschitz extensions can often be uncomputable. In addition, similar to smooth sensitivity, we achieve optimality and NP-hardness guarantees for accuracy in this generalized setting under several reasonable metrics of optimality.

Furthermore, our Sensitivity-Preprocessing Function only requires a simple recursive construction that is more conducive to efficient implementation, which we achieve for important statistical functions such as median, mean, and variance. These functions have been of particular interest for similar techniques because they are highly important statistics and also have large worst-case sensitivity, but small average sensitivity. We will compare our results to previous results in the following section, where our framework gives state-of-the-art accuracy for median and mean when no underlying assumptions are made to the database. The key assumption that we would aim to avoid is that data points are drawn iid, which is a popular assumption in previous results for outputting functions such as α -trimmed mean (i.e., Propose-Test-Release [DL09]). While this assumption is quite standard, we contend that real world data are often not iid, and so consideration of the more general case is still an important problem. Comparing our framework to those that apply the iid assumptions (and sometimes further assumptions, such as being drawn from a Gaussian [KLSU]), we will also achieve similarly high accuracy for well concentrated databases, but concede that mechanisms specifically catered for that setting will often be superior. However, we note that because our framework is a preprocessing routine, it can be run before applying any differentially private mechanism such as Propose-Test-Release to achieve further gains in accuracy. In avoiding this iid assumption, the primary technique we will then compare our framework with will be smooth sensitivity which is popularly used for privately outputting median. Both frameworks have database-specific accuracy and direct comparison will be difficult, but we give strong evidence in the next section of why our framework compares favorably to smooth sensitivity for median.

The localized construction of our Sensitivity-Preprocessing Function will also allow us to tailor the new sensitivity parameters beyond previous techniques. To this end, we introduce a more refined sensitivity metric, which we call *individual sensitivity*, and show that it is important for a variant definition of *personalized differential privacy* introduced in [GR15], and used in subsequent works on market design for private data [CCK⁺13, CLR⁺15]. We can apply our construction as a preprocessing step for more refined sensitivity tailoring to take advantage of personalized differential privacy guarantees. We believe this application of our results may be of independent interest for future work in these directions.

In this work we cover a broad range of the more immediate results from this new framework, but believe that there is still a substantial amount of work in this direction. While some of our proofs will become involved, all of our results follow from first principles, suggesting the potential for further results using more sophisticated tools within this framework. These further results

include: efficient implementations of more difficult functions such as linear regression; optimizing the trade-off between decreasing the sensitivity parameter and the error incurred by our Sensitivity-Preprocessing for specific functions; applying our algorithm in the markets for privacy literature; and variants of our algorithm that are optimized for specific computational settings or application domains.

1.1 Our Results

Our results will primarily revolve around the *Sensitivity-Preprocessing Function*, which we introduce below. It is an alternate schema for fitting a general function to a sensitivity-bounded function in the context of differential privacy. More specifically, we consider the general problem of taking any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and constructing a new function $g : \mathcal{D} \rightarrow \mathbb{R}$ that satisfies given sensitivity parameters and minimizes the difference $|f(D) - g(D)|$ over all databases $D \in \mathcal{D}$. The sensitivity parameters we consider will be more refined and we define *individual sensitivity*, which is the maximum change in a function’s value from adding or removing a single specific data entry. We use Δ_i to denote individual sensitivity to the i -th data entry. When a database is comprised of data from multiple individuals, Δ_i captures the sensitivity of the function to person i ’s data.

In this section, we first give an overview of our recursively constructed Sensitivity-Preprocessing Function that works in a highly generalized setting, along with the corresponding runtime and error guarantees. We then examine the optimality and hardness of this general function in the context of minimizing $|f(D) - g(D)|$ over all databases $D \in \mathcal{D}$. While constructing our Sensitivity-Preprocessing Function will require exponential time in general, we show that it can be simply and efficiently implemented in $O(n)$ time for median (with presorted data), and in $O(n^2)$ time for several other important statistical measures including mean and variance. We show that our Sensitivity-Preprocessing Function tailors an important metric (individual sensitivity) in the variant definition of *personalized differential privacy*, which provides different privacy guarantees to different individuals in the same database, and is a useful tool in the design of markets for privacy. We further generalize our construction of Sensitivity-Preprocessing Function to bound the ℓ_1 sensitivity of 2-dimensional functions $f : \mathcal{D} \rightarrow \mathbb{R}^2$, and show that such techniques cannot be extended to higher dimensions without treating each dimension independently.

Sensitivity-preprocessing function overview

Our construction of the Sensitivity-Preprocessing Function is similar to the Lipschitz extension framework, but we extend only from the empty set. We start with $f(\emptyset) = g(\emptyset)$, and inductively construct g for larger databases while trying to achieve two desiderata: (1) maintain the appropriate individual sensitivity bounds; and (2) keep g as close as possible to f . The first objective will be strictly maintained, and we will optimize over the second objective.

The primary difficulty in this construction is that we often consider \mathcal{D} to be infinite. As a result, checking to make sure we do not violate any sensitivity constraints when defining g on a new database can require checking all databases on which g was previously defined. For example, general Lipschitz extensions require checking all previously defined databases to extend to another database, which can often be uncomputable for general functions. To avoid these uncomputability issues, we take advantage of the lattice structure of neighboring databases in the differential privacy landscape. This will allow us to give a far more localized construction that critically utilizes the following two key properties of the data universe metric space:

1. While each database could have infinitely many neighboring databases, it only has a finite number of neighbors with strictly fewer entries.

2. Any two neighbors of a strictly larger database must also be neighbors of a strictly smaller database.

These properties ensure that whenever we define g on a new database, we only need to check that sensitivity constraints of strictly smaller neighboring databases are satisfied. Once we have found the feasible range of g that does not violate any sensitivity constraints, we will define g to be as close as possible to f within this feasible range.

Definition 1.1 (Informal version of Definition 3.1). Given a function $f : \mathcal{D} \rightarrow \mathbb{R}$ and fixed sensitivity parameters, we recursively define our Sensitivity-Preprocessing Function $g : \mathcal{D} \rightarrow \mathbb{R}$ such that $g(\emptyset) = f(\emptyset)$ ¹ and for any $D \in \mathcal{D}$,

$$g(D) = \text{closest point to } f(D) \text{ in } \text{FEASIBLE}(D),$$

where $\text{FEASIBLE}(D)$ is the set of all points that do not violate the sensitivity constraints based upon $g(D')$ for all neighbors D' of D with fewer entries.

The recursive structure of this function allows us to compute $g(D)$ by only looking at the subsets of D , which unfortunately takes exponential time. Later in the paper (Sections 5 and 6), we utilize the simplicity of the recursive structure to efficiently implement this algorithm for several functions of interest that exhibit additional structure. Theorem 1.2 summarizes our main result on the running time and accuracy guarantees of our Sensitivity-Preprocessing Function algorithm.

Theorem 1.2 (Informal version of Theorem 3.3). *For any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and desired sensitivity bounds $\{\Delta_i\}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f . Given query access to f in $T(n)$ time for a database of size n , we give $O((T(n) + n)2^n)$ time access to $g(D)$ for any $D \in \mathcal{D}$ with n entries. We also give instance-specific bounds on each $|f(D) - g(D)|$ based on the sensitivity of f and $\{\Delta_i\}$.*

Our algorithm for computing the Sensitivity-Preprocessing Function g (Algorithm 1) is robust to informational assumptions. We only assume query access to f , and do not require any knowledge of the database universe \mathcal{D} or the sensitivity of f .

This algorithm easily extends to functions that map to \mathbb{R}^d , by treating each dimension independently. See Remark 3.4 and Section 8 for more details on handling high-dimensional functions.

Approximate Optimality and Hardness

The construction of our Sensitivity-Preprocessing Function is quite simple in its greedy structure and requires exponential running time. To justify these two properties, we complement our algorithm with both optimality guarantees and hardness results.

In particular, we still consider the general problem of taking any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and constructing a new function $g : \mathcal{D} \rightarrow \mathbb{R}$ with individual sensitivity bounds $\{\Delta_i\}$. The goal will then be to minimize the difference $|f(D) - g(D)|$ across databases $D \in \mathcal{D}$. Despite its simplicity, we show that our Sensitivity-Preprocessing Function still achieves a 2-approximation to the optimal function in the ℓ_∞ metric in this generalized setting.

Proposition 1.3 (Informal version of Corollary 4.4). *Given any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and sensitivity parameters $\{\Delta_i\}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be our Sensitivity-Preprocessing Function. For any function $f^* : \mathcal{D} \rightarrow \mathbb{R}$ with individual sensitivity bounds $\{\Delta_i\}$,*

$$\max_{D \in \mathcal{D}} |f(D) - g(D)| \leq 2 \max_{D \in \mathcal{D}} |f(D) - f^*(D)|.$$

¹See Remark 3.2 for a discussion of how to initialize $g(\emptyset)$ if $f(\emptyset)$ is not well-defined.

Our guarantees are even stronger because they also hold over finite subsets of the data universe. While Proposition 1.3 measures error in the worst-case over \mathcal{D} , we also show (Lemma 4.2) that when the optimal error is small on certain subsets of the data universe, then our error is also small.

Furthermore, we can show that our Sensitivity-Preprocessing Function is Pareto optimal: there is no strictly superior sensitivity-bounded function that improves accuracy over all databases.

Proposition 1.4 (Informal version of Lemma 4.5). *Given any $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with individual sensitivity parameters $\{\Delta_i\}$. For any $f^* : \mathcal{D} \rightarrow \mathbb{R}$ with individual sensitivity parameters $\{\Delta_i\}$, if there is some $D \in \mathcal{D}$ such that*

$$|f(D) - f^*(D)| < |f(D) - g(D)|,$$

then there also exists some $D' \in \mathcal{D}$ such that

$$|f(D') - f^*(D')| > |f(D') - g(D')|.$$

These results imply that our Sensitivity-Preprocessing Function does quite well fitting to the original function under the metrics we are considering. However, it does take exponential time, so we complement these results by showing that getting the same approximation guarantees is NP-hard even for a single sensitivity parameter $\{\Delta_i\} = \Delta$.

Proposition 1.5 (Informal version of Proposition 4.6). *Given any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and sensitivity parameter Δ , it is NP-hard to construct any function $f^* : \mathcal{D} \rightarrow \mathbb{R}$ with sensitivity Δ that enjoys the same accuracy guarantees as our Sensitivity-Preprocessing Function.*

We further argue that it is uncomputable to do better than a 2-approximation in the ℓ_∞ metric, and also uncomputable to achieve even a constant approximation in any ℓ_p metric for $p < \infty$, which justifies our choice of metric. We believe that the combination of these results gives a strong indication that our Sensitivity-Preprocessing Function and corresponding exponential time construction is the best we can hope to achieve for the general problem under reasonable metrics of optimality.

Efficient Implementation for Important Statistical Measures

One of the main benefits of our Sensitivity-Preprocessing Function is that its simple recursive structure is conducive to giving simple efficient variants for specific functions through largely straightforward state space reductions and dynamic programming. To this end, we give efficient implementations of our Sensitivity-Preprocessing Function for the important statistical functions mean, median, α -trimmed mean, maximum, minimum, and variance. These statistical metrics can be surprisingly difficult to release privately without assuming the input is restricted to some range, and often requires further assumptions for metrics like mean, such as data being drawn from identical and independent distributions. In fact, for Propose-Test-Release even the iid assumption is not sufficient to apply their framework to mean, and other works required further concentration properties such as being drawn from the normal distribution. However, our Sensitivity-Preprocessing Function does not require bounded sensitivity of the input function f , and can also avoid any iid requirements. As a result, we are able to efficiently implement each of these statistical metrics with no constraints on the inputs. It is important to note that our implementations only consider a single sensitivity parameter Δ , but we believe each can be efficiently extended for individual sensitivity parameters $\{\Delta_i\}$.

For each of these statistical metrics, we are simply implementing our Sensitivity-Preprocessing Function more efficiently, so all of the previously stated optimality guarantees still apply. To further

strengthen these optimality guarantees, we give a more rigorous treatment of the error incurred by our efficient implementation of median, mean, and variance, which we consider to be three of the most fundamental statistical tasks.

Median We focus on privately and accurately computing median, because it has been extensively studied under smooth sensitivity [NRS07]. Both our framework and smooth sensitivity provide database-specific accuracy guarantees, so a direct comparison of accuracy will be difficult. Nevertheless, we show that our framework compares favorably to smooth sensitivity on median.

We begin by stating our result which will use a definition from [NRS07] to give a more apparent comparison in terms of accuracy. As in [NRS07], we formally define

$$A^{(k)}(D) = \max_{d(D,D') \leq k} LS_f(D')$$

as the k -local sensitivity of a function f for database D . For median this just reduces to $A^{(k)}(D) = \max_{0 \leq t \leq k+1} (x_{m+t} - x_{m+t-k-1})$ where $m = \frac{n+1}{2}$ (when n is assumed to be odd). When n is even, we define the median to be the average of the middle two entries (i.e. $(x_{\lfloor \frac{n+1}{2} \rfloor} + x_{\lceil \frac{n+1}{2} \rceil})/2$) as is standard, and the resulting interpretation of $A^{(k)}(D)$ is nearly identical.

We also need to define median on the empty set. Since this is not naturally defined, we allow it to be an input parameter $med(\emptyset)$ chosen by the data analyst as the estimated median. As our comparison will mostly be with smooth sensitivity which must assume values are in a bounded range $[min, max]$, the natural choice would be $med(\emptyset) = \frac{max-min}{2}$. Further, it would be natural in this setting to set our parameter $\Delta = \frac{max-min}{n}$.

Theorem 1.6. *Let $med : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the median function for the data universe of all finite-length real-valued vectors. For chosen parameters $med(\emptyset)$ and Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, if $x_1 \leq \dots \leq x_n$ we give $O(n)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that $g(D) = med(D)$ whenever $A^{(k)}(D) \leq 2(k+1)\Delta$ for $k \leq n/4$ and $med(D) \in [med(\emptyset) - \frac{n}{2}\Delta, med(\emptyset) + \frac{n}{2}\Delta]$.*

To interpret this accuracy result, we begin by comparing performance on the specific example that considered by [NRS07], where smooth sensitivity performed well. In fact, it was exactly this setting that motivated our choice of assumptions under which to show our framework outperforms smooth sensitivity. Consider an environment where data points x_1, \dots, x_n lie in a bounded range $[0, 1]$, and we naturally set $med(\emptyset) = 1/2$ and $\Delta = 1/n$. Consider the particular database $D = (x_1, \dots, x_n)$, where $x_i = i/n$. In this example, it is easy to check that the assumptions are satisfied for our Sensitivity-Preprocessing Function to correctly output $g(D) = med(D)$. Privately answering the query $g(D)$ using the Laplace mechanism (see Definition 7.2 for a formal definition) outputs $med(D)$ plus noise with scale $\Delta/\epsilon = \frac{1}{\epsilon n}$. In contrast the noise parameter added under smooth sensitivity (without our sensitivity preprocessing) would have to scale $\frac{1}{\epsilon^2 n}$, which is asymptotically larger, and would thus yield significantly lower accuracy.

The assumptions in Theorem 1.6 that $A^{(k)}(D) \leq 2(k+1)\Delta$ for all $k \leq n/4$ and $med(D) \in [med(\emptyset) - \frac{n}{2}\Delta, med(\emptyset) + \frac{n}{2}\Delta]$ are then exactly the generalization of this condition where our database still has values that are reasonably spread out, but also has low local sensitivity and we would still like to achieve high accuracy. Further note that these assumptions allow both the bottom and top quartile values to be arbitrarily small and large, implying that our construction is able to handle outliers well. Restricting our attention to databases that satisfy these conditions allows us to consider all of the databases under which smooth sensitivity performs well. Under these assumptions, smooth sensitivity will achieve a noise magnitude of $\frac{\Delta}{\epsilon^2}$, whereas we instead achieve

an asymptotically better noise magnitude of $\frac{\Delta}{\epsilon}$. Note that smooth sensitivity requires a bounded range, which we are considering here to be of size $n\Delta$, giving a global sensitivity of $n\Delta$ for median. Accordingly, standard mechanisms would have noise magnitude of $\frac{n\Delta}{\epsilon}$, which is significantly worse than both our framework and smooth sensitivity.

It is important to acknowledge that our Sensitivity-Preprocessing framework will not outperform smooth sensitivity in general. For example, consider again the domain where all data points are bounded in $[0, 1]$, and consider the database D of all 1's. Then our $g(D) = 1$, and the Laplace Mechanism would output $med(D)$ with noise of magnitude $\frac{1}{\epsilon n}$. However, the smooth sensitivity of this database will be $e^{-\epsilon n/2}$, and the smooth sensitivity framework only requires noise of magnitude $e^{-\epsilon n/2}/\epsilon$. More generally, smooth sensitivity will often do better if most data entries are exponentially close to one value ². To achieve benefits from both techniques, an analyst could simply apply the smooth sensitivity framework after our preprocessing step. Our preprocessing algorithm will only improve the smooth sensitivity parameters, so this approach will continue to achieve the strong accuracy guarantees of smooth sensitivity on highly concentrated databases ³. On the example above, if we first apply our Sensitivity-Preprocessing Function and then add noise based on the smooth sensitivity, then we will also have noise that scales approximately as $e^{-\epsilon n/2}/\epsilon$. Since our algorithm is a preprocessing step, it is compatible with all techniques for improving accuracy of differentially private algorithms. We view this as an exciting avenue for future work, to optimize the use of each tool under different parameter settings.

Mean. We first note that mean is not naturally defined on the empty set, so we define it to be an input parameter $\hat{\mu}$ chosen by the data analyst as the estimated mean. The analyst's choice of $\hat{\mu}$ should reflect her prior knowledge, and will play a role in our accuracy guarantees. Intuitively, if two databases have means that are exponentially far apart, we cannot hope to output both means accurately. As such, our Sensitivity-Preprocessing Function will be accurate on databases with mean reasonably close to $\hat{\mu}$. Our efficient implementation of mean will take $O(n^2)$ time and provide the following guarantees.

Theorem 1.7. *Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function for the data universe of all finite-length real-valued vectors. For chosen parameters $\hat{\mu}$ and Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, we give $O(n^2)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that,*

$$|g(D) - \mu(D)| \leq \max \left\{ |\mu(D) - \hat{\mu}| - \frac{n}{3}\Delta, 0 \right\} + \sum_{i=1}^n \max \left\{ \frac{27|x_i - \mu(D)|}{n} - \Delta, 0 \right\}.$$

Additionally, if we are guaranteed that each $x_i \in [\hat{\mu} + \alpha\Delta, \hat{\mu} + (\alpha + n)\Delta]$ for $\alpha \in [-n, 0]$, then $g(D) = \mu(D)$

As was previously mentioned, we claim that our framework gives state-of-the-art accuracy for privately outputting mean when no underlying assumptions are made on the database. It turns

²It is also necessary to mention neither mechanism will necessarily outperform the other when the assumptions are not fulfilled, and for databases with high local sensitivity both mechanisms will have poor accuracy but in different ways. Consider again the domain where all data points are bounded in $[0, 1]$, and let the database D consist of $\frac{n+1}{2}$ values 1 and $\frac{n-1}{2}$ values 0, so $med(D) = 1$. Our Sensitivity-Preprocessing Function will output $g(D) \approx 1/2$ and add noise with magnitude $\frac{1}{\epsilon n}$, while smooth sensitivity will add prohibitively large noise parameter of scale $1/\epsilon$ to $med(D)$, and neither gives any accurate information on the true median value.

³We note that this can be done efficiently, as both smooth sensitivity and our preprocessing step take time $O(n^2)$ on database-ordered functions. Database-ordered functions will be defined in Section 5, and it will be seen that this general class can be implemented in $O(n^2)$ time for our framework. While it is outside the scope of this paper, it is straightforward to see that this also holds for the smooth sensitivity framework and that this property is preserved when applying our preprocessing to the median function. We leave formal proofs of these to future work.

out that the lack of assumptions on the database makes outputting mean privately incredibly difficult, where even Propose-Test-Release was unable to privately output mean with iid assumptions. Often further assumptions such as data drawn from a normal distribution or other distributions that concentrate well are necessary to guarantee highly-accurate private output of mean. To our knowledge, the best algorithm to output mean privately when no underlying assumptions are made is the naive algorithm, that simply considers the range $[\hat{\mu} - \frac{n}{2}\Delta, \hat{\mu} + \frac{n}{2}\Delta]$ of length $n\Delta$, and rounds up or down any value outside of this range. It is important to note that this range must be chosen independently of the database, as catering the range to the considered database can easily be shown to violate privacy. Restricting values to a range of $n\Delta$ will then ensure that global sensitivity is at most Δ and standard mechanisms can be applied from here. For all databases with values inside the range $[\hat{\mu} - \frac{n}{2}\Delta, \hat{\mu} + \frac{n}{2}\Delta]$ this will then give accurate output.

Note that our second accuracy guarantee similarly considers databases under which our preprocessing correctly outputs the mean. It can be immediately seen that the allowable range for values in the database extends beyond the range for the naive algorithm, and can in some ways be seen to double this range. Essentially, the minimum and maximum values must still be within $n\Delta$ for us to guarantee correctly outputting the mean, but the range under which minimum and maximum values can fall is now doubled. Given the significance of mean as a statistical metric, we still believe this improvement is of significance and is the first to improve upon the naive algorithm when no underlying assumptions are made with regard to the database.

To complement this comparison to the naive algorithm, we also give strong accuracy guarantees for all databases not just the ones output correctly in our preprocessing. Unpacking the bound in Theorem 1.7, the first term says that the mean of the database cannot be too far from $\hat{\mu}$. The second term considers the individual sensitivity of each data point, where $|x_i - \mu(D)|/n$ is roughly the amount the mean changes from adding x_i to the database. The sensitivity bound on g requires that each individual change can only be offset by an additive Δ , and we need to consider this contribution from each input. Intuitively, our error is small for databases whose mean is reasonably close to $\hat{\mu}$ and do not have many significant outliers, which is exactly what one would expect.

Variance. Variance is also not naturally defined on the empty set, so we define it to be 0 for simplicity. Our efficient implementation of variance takes $O(n^2)$ time and provides the following guarantee.

Theorem 1.8. *Let $\mathbf{Var} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the variance function for the data universe of all finite-length real-valued vectors. For fixed parameter Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, we have $O(n^2)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that,*

$$|g(D) - \mathbf{Var}[D]| \leq \max\left\{\mathbf{Var}[D] - \frac{n}{2}\Delta, 0\right\} + \sum_{i=1}^n \max\left\{\sum_{j=1}^n \frac{4(x_i - x_j)^2}{n^2} - \Delta, 0\right\}.$$

The primary takeaway from the bound in Theorem 1.8 is that databases with reasonably small variance and no major outliers will have low error bounds. The first term in the error bound says that the variance of our database cannot be too large, which follows from our choice of the empty set to be defined at 0. The second term is a bit more messy, but has a natural interpretation under the known deformation of variance, where we can consider $\sum_{j=1}^n (x_i - x_j)^2/n^2$ to be the contribution of input x_i to the variance. This contribution can then be offset by Δ , and we need to consider this contribution from each input.

Personalized Privacy

Due to the preprocessing aspect of our Sensitivity-Preprocessing Function, we can also apply our framework to variant definitions of differential privacy. In particular, we consider *personalized differential privacy* introduced in [GR15], which allows for a more refined definition whereby each individual receives their own ϵ_i privacy parameter. Our definition of *individual sensitivity* is then motivated by this privacy variant, as it can be exactly seen as the complementary sensitivity measure for this variant. More specifically, most privacy mechanisms add noise proportional to $\Delta f/\epsilon$ for outputting functions $f : \mathcal{D} \rightarrow \mathcal{R}$ while still preserving ϵ -differential privacy. This intimate connection between Δf and ϵ in the output accuracy will be equivalent for the individual sensitivity measures $\Delta_i(f)$ and its respective ϵ_i . Consequently, the necessary noise for personalized privacy will be proportional to $\max_i \Delta_i(f)/\epsilon_i$. We will formally prove this fact for two of the most fundamental mechanisms, Laplace and Exponential, and further remark (Remark 7.6) that this approach extends to any ϵ -differentially private mechanism.

Theorem 1.9 (Informal version of Propositions 7.3 and 7.5). *For both the Laplace and Exponential Mechanisms, instead of adding noise proportional to Δ/ϵ , the added noise can be proportional to $\max_i \Delta_i/\epsilon_i$ to ensure personalized differential privacy for privacy parameters $\{\epsilon_i\}$.*

As a result, it is no longer necessarily optimal to set the individual sensitivity parameters $\{\Delta_i\}$ in our Sensitivity-Preprocessing Function to be equal, but instead set them according to the given $\{\epsilon_i\}$ privacy parameters towards the goal of having each $\Delta_i(g)/\epsilon_i$ roughly equal. This extends the interest in our Sensitivity-Preprocessing Function beyond the context of dealing with worst-case sensitivity being much greater than average sensitivity.

For example, consider a well-behaved function where $\{\Delta_i\} = \Delta$, and $\{\epsilon_i\} = \epsilon$ for all i except for some individual j where $\epsilon_j = \epsilon/2$. Under this situation it may instead be optimal to halve the individual sensitivity of j , which will allow adding half as much noise while only incurring a small additive error by restricting the sensitivity of just one person.

In addition, the error bounds from our general procedure allow for the intuitive fact that increasing any individual sensitivity will increase the accuracy of our preprocessing step for databases including that individual. We note that when trying to preserve the fraction $\Delta_i(g)/\epsilon_i$, any increase in ϵ_i (reduced privacy for individual i) will allow us to increase our Δ_i parameter, improving accuracy as desired. In this way, our Sensitivity-Preprocessing Function is able to fully take advantage of the heterogeneous ϵ_i in the variant definition of personalized privacy, and is the first to give accuracy bounds that increase/decrease independently with respect to each ϵ_i .

We believe that these accuracy guarantees can be of further interest in the context of markets for privacy, where individuals sell their data to an analyst and demand different amounts of privacy, represented by their respective ϵ_i . The trade-off between privacy and accuracy is naturally formalized in these markets through the analyst's budget for procuring accurate estimates of population statistics. Applying our Sensitivity-Preprocessing Function to achieve individualized privacy guarantees will allow the analyst to more optimally balance these trade-offs because the accuracy will respond proportionally to changes in privacy for each individual.

Higher-Dimensional Extensions for ℓ_1 Sensitivity

Our Sensitivity-Preprocessing Function was only defined for 1-dimensional $f : \mathcal{D} \rightarrow \mathbb{R}$. We also consider the setting where $f : \mathcal{D} \rightarrow \mathbb{R}^d$. We note that our Sensitivity-Preprocessing Function could instead be given parameters $\{\Delta_i\}$ where $\Delta_i = (\Delta_{i,1}, \dots, \Delta_{i,d})$ has different sensitivity parameters for each dimension of the function. We could then apply our Sensitivity-Preprocessing Function to each

dimension independently and would achieve the corresponding bounds on sensitivity. However, this approach would require adding noise independently to each dimension when applying a differentially private mechanism on the Sensitivity-Preprocessing Function. Instead, we would like to only require bounds on the ℓ_1 sensitivity of our constructed function.

We give a natural extension of our Sensitivity-Preprocessing Function to higher dimensions, and show that the accuracy guarantees continue to hold in ℓ_1 -distance when f is 2-dimensional (Theorem 8.3). We also show that this construction fails to extend to higher dimensions because a key fact about the intersection of ℓ_1 balls only holds in 1 and 2 dimensions.

1.2 Related Work

Our work touches upon several areas of interest. We first discuss previous work on dealing with outlying databases within the data universe, and then discuss previous work on personalized differential privacy and its use within the markets for privacy literature.

Worst-case vs average-case sensitivity

Instance-specific noise for dealing with worst-case sensitivity was first introduced in [NRS07], where they considered adding noise proportional to *local sensitivity*. In order to avoid leaking too much information through noise added by local sensitivity, [NRS07] constructed a *smooth sensitivity* metric that minimized the instance-specific noise while still ensuring differential privacy. They further showed that smooth sensitivity could be efficiently computed and utilized for a variety of important functions for which average sensitivity was much smaller than *global sensitivity*. However, for some functions computing smooth sensitivity was NP-hard or even uncomputable, which inspired the introduction of Sample-and-Aggregate, a technique that preserved privacy and was efficient on all functions with bounded range and for sufficiently large databases. The general idea was to approximate the function with random subsets of the given database in order to impose stronger bounds on the sensitivity of this approximation. Combining this with smooth sensitivity allowed for strong error guarantees under the assumption that random subsets of the database often well-approximated the full database.

In order to avoid some of these assumptions, an alternate framework, Propose-Test-Release, was provided in [DL09], which also heavily relied on the notion of local sensitivity. In particular, their framework would check if a given database was “far away” from an outlier, and only release an accurate estimate of the output under this specific circumstance, while outputting \perp (meaning NULL) otherwise. Furthermore, this algorithm would define the outlying databases by explicitly setting the allowable upper bound on local sensitivity. They show how to implement this framework efficiently for several important functions, and give strong error guarantees when the mechanism does not output \perp .

Both of these frameworks relied upon local sensitivity, which is still a worst-case metric. It is possible for most databases to have high local sensitivity while still having small average sensitivity. To remedy this issue, previous work instead considered fitting the original function to one with global sensitivity closer to the average sensitivity. This preprocessing step can largely be thought of as forcing the output of outlying databases to be closer to that of the well-behaved databases. This procedure then fits in the general notion of Lipschitz extensions. Informally, Lipschitz extensions show that there always exists an extension of a smooth function restricted to a subspace to the entire metric space. By considering “smoothness” in the context of differential privacy to be the sensitivity of the function, previous work generally considers the restricted subspace to be the well-behaved databases.

Lipschitz extensions were first implicitly used in [KNRS13] under the context of *node* differential privacy. This work considered restricting the maximum degree of graphs for outputting a variety of graph statistics in bounded-degree graphs. This work was then extended in [RS16] which gave efficient Lipschitz extensions for higher-dimensional functions on graphs such as degree distribution. In this work, [RS16] further utilize Lipschitz extensions for a generalization of the exponential mechanism. Lipschitz extensions were also considered in [BBDS13] where the goal was to achieve a *restricted sensitivity* under a certain hypothesis of the database universe and extending to the entire data universe with this global sensitivity constraint. While this procedure was in general computationally inefficient, [BBDS13] gave efficient versions for subgraph counting queries and local profile queries.

Our technique of considering only strictly smaller neighboring databases is related to a technique used to achieve differential privacy over graphs. The *down sensitivity* [RS] (also called *empirical global sensitivity* in [CZ13]) of a function at a graph G is the global sensitivity of the function when restricted to the space of all subgraphs of G . That is, it is the maximum change in the function’s value between any two neighboring subgraphs of G . Similar to our work, this requires checking sensitivity on a smaller number of neighboring databases, and can allow less noise to be added to analysis on databases with small down sensitivity. Through this lens, our construction of Sensitivity-Preprocessing Function can be viewed as ensuring that all databases have low down sensitivity. However, an important distinction between these two results is that down sensitivity considers all pairs of neighboring subgraphs of G , which, for example, may be the empty graph and a single node for a large graph G . To contrast, at each recursive step of our algorithm, we only consider only smaller neighbors of the current database, i.e., with one entry removed. This refined analysis means that a database might have large down sensitivity, and our Sensitivity-Preprocessing Function can still be accurate

Personalized privacy and markets for privacy

We show how our Sensitivity-Preprocessing framework can be applied to *personalized differential privacy*, where each user in the database has her own privacy parameter ϵ_i . This definition was first introduced by [GR15], in the context of purchasing data from privacy-sensitive individuals. A subsequent line of work on market design for private data [CCK⁺13, NOS12, NST12, LR12, FL12, GR15, GLRS14, CLR⁺15, CIL15, WFA15, CPWV16] leveraged personalized privacy guarantees to purchase data with different privacy guarantees from individuals with heterogeneous privacy preferences. The vast majority of this work focused on the market design problem of procuring data, and not on the differentially private algorithms that provided personalized privacy guarantees. [CLR⁺15] gave a technique for achieving personalized privacy for linear functions by reweighting each person’s data inversely proportional to their privacy guarantee. Unfortunately, this reweighting technique does not extend beyond linear functions. [CCK⁺13] proposed an even stronger notion of personalized privacy, that was both personalized and data-dependent, but did not give any algorithmic techniques to satisfy this definition.

Several other results gave mechanisms specific to personalized differential privacy by randomly keeping each individual’s data in the database with probability proportional to their respective ϵ_i [JYC15, AGK17, LXJJ17]. However, they are unable to provide corresponding error guarantees with such procedures for general functions. [AKZ⁺17] gave a technique for providing two-tiered personalized privacy guarantees. Some users received differential privacy and some users received a stronger guarantee of *local differential privacy*, where the users do not trust the data analyst to see their true data.

Finally, there is a small body work on high probability privacy guarantees and average-case

privacy guarantees [BLR08, CLN⁺16, BF16]. This work addresses a very different problem than we study here. These papers assume that databases are sampled according to some distribution over the data universe, and provide high probability guarantees with respect to the sampling distribution, allowing a failure of either privacy or accuracy on some set of unlikely databases. To contrast, we assume that databases are fixed, not randomly sampled. We provide privacy and accuracy guarantees that depend on the well-behavedness of a given function over the data universe, and our guarantees hold everywhere in the data universe.

1.3 Organization

In Section 2, we introduce some of the notation and basic definitions that will be used throughout the paper. In Section 3, we introduce our Sensitivity-Preprocessing Function and prove its general accuracy guarantees. In Section 4, we give optimality and hardness guarantees for our sensitivity-preprocessing procedure. In Section 5, we show that several important functions can be efficiently implemented in our framework, such as mean, median, maximum, minimum, and we also give strong error guarantees on the implementation of mean. In Section 6, we efficiently implement our framework for variance and give corresponding error guarantees. In Section 7, we prove several useful facts regarding individual sensitivity and the variant definition of personalized differential privacy, and show how our framework can be very useful in this context. In Section 8, we consider a natural extension of our algorithm that bounds a function’s sensitivity in the ℓ_1 metric for 2 dimensions.

2 Preliminaries

We introduce the standard notion of differential privacy and the corresponding global sensitivity metric. We say that two databases are *neighboring* if they differ in at most one entry.

Definition 2.1 (Differential privacy [DMNS06]). A mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ is ϵ -*differentially private* if for every pair of neighboring databases $D, D' \in \mathcal{D}$, and for every subset of possible outputs $\mathcal{S} \subseteq \mathcal{R}$,

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in \mathcal{S}].$$

Definition 2.2 (Global Sensitivity). The *global sensitivity* of a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$ is:

$$\Delta f = \max_{D, D', \text{ neighbors}} \|f(D) - f(D')\|_1.$$

Our result is primarily concerned with tailoring a more refined version of global sensitivity, for which we will need more specific notation for neighboring databases. In particular, we will consider the data universe \mathcal{D} to be composed of (a possibly infinite) collection of individuals, where x_i will denote the data of individual i . Any database $D \in \mathcal{D}$ is then composed of the data of some finite subset of individuals I , so $D = \{x_i : i \in I\}$. For ease of notation, we will often assume that $D = (x_1, \dots, x_n)$. Further, if individual i ’s data is contained in database D , then we will consider $D - x_i$ to be the database with individual i ’s data removed. Similarly, if i ’s data is not included in database D , then we consider the database $D + x_i$ to be the database with i ’s data added. We will also sometimes use $i \in D$ to denote that individual i ’s data is included in database D . Finally, we also assume that for any $D \in \mathcal{D}$, if $D' \subset D$ is non-empty, then $D' \in \mathcal{D}$.

With this notation, we introduce the notion of individual sensitivity that is the maximum change in output that is possible by adding individual i ’s data.

Definition 2.3 (Individual Sensitivity). The *individual sensitivity* of a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$ with respect to i is:

$$\Delta_i(f) \stackrel{\text{def}}{=} \max_{x_i, \{D: i \notin D\}} \|f(D) - f(D + x_i)\|_1.$$

We further let $\{\Delta_i(f)\}$ denote the individual sensitivities of f to all individuals.

For reference, we also provide the definition of local sensitivity that will not be used in this work, but was referred to extensively in related works.

Definition 2.4 (Local Sensitivity). The *local sensitivity* of a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$ at database $D \in \mathcal{D}$ is:

$$\Delta_D f = \max_{D': \text{neighbor of } D} \|f(D) - f(D')\|_1.$$

3 Sensitivity-Preprocessing Function

In this section we formally define our Sensitivity-Preprocessing Function, give the corresponding constructive algorithm for accessing this function, and prove instance-specific error bounds between the original function and our Sensitivity-Preprocessing Function. Recall that our primary goal is to give an alternate schema for fitting a general function to a sensitivity bounded function. More specifically, suppose we are given a function $f : \mathcal{D} \rightarrow \mathbb{R}$ and desired sensitivity parameters $\{\Delta_i\}$, and want to produce another function $g : \mathcal{D} \rightarrow \mathbb{R}$ that closely approximates f , and has individual sensitivity at most Δ_i for all i .

The Sensitivity-Preprocessing Function will ultimately be defined as a simple greedy recursion that builds up from the empty set. The key insight is that we can take advantage of the particular metric space structure of databases such that defining our function on a new database only depends on the subsets of that database. We first use the fact that while each database could have infinitely many neighboring databases, it only has a finite amount of neighbors with strictly fewer entries. This will allow us to only consider the constraints incurred by each $D - x_i$ for some database D . In particular, for each $g(D - x_i)$ it is allowable to place $g(D)$ anywhere in the region $[g(D - x_i) - \Delta_i, g(D - x_i) + \Delta_i]$. Intersecting each of these intervals will give the feasible region for $g(D)$, and we will greedily chose the point closest to $f(D)$. We then use the fact that any two neighbors of a strictly larger database must also be neighbors of a strictly smaller database. This will ensure that the intersection of all feasible intervals is non-empty, even under our greedy construction.

As a result, the Sensitivity-Preprocessing Function g is defined inductively starting from the empty set, and new data points are added one by one. The algorithm ensures that the value of g changes by at most Δ_i when new data point x_i is added, while minimizing the distance $|f(D) - g(D)|$ at every point.

Definition 3.1 (Sensitivity-Preprocessing Function). Given any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and non-negative parameters $\{\Delta_i\}$, we say that a function $g : \mathcal{D} \rightarrow \mathbb{R}$ is a Sensitivity-Preprocessing Function of f with parameters $\{\Delta_i\}$, if $g(\emptyset) = f(\emptyset)$ ⁴ and

$$g(D) = \begin{cases} \text{UPPER}(D), & \text{if } \text{UPPER}(D) \leq f(D) \\ \text{LOWER}(D), & \text{if } \text{LOWER}(D) \geq f(D) \\ f(D), & \text{otherwise} \end{cases}$$

where $\text{UPPER}(D) = \min_{j \in D} \{g(D - x_j) + \Delta_j\}$ and $\text{LOWER}(D) = \max_{j \in D} \{g(D - x_j) - \Delta_j\}$.

⁴See Remark 3.2 for a discussion of how to initialize $g(\emptyset)$ if $f(\emptyset)$ is not well-defined.

If $\{\Delta_i\} = \Delta$ for some non-negative Δ , then we say that g is a Sensitivity-Preprocessing Function of f with parameter Δ .

The generalization from global sensitivity to individual sensitivities is critical to our personalized privacy results in Section 7. This generalization does not increase the running time, and it is easy to see that this yields global sensitivity equal to the maximum individual sensitivity.

3.1 Algorithmic Construction of Sensitivity-Preprocessing Function

The algorithm PREPROCESSING is presented in Algorithm 1. It begins by initializing $g(\emptyset) = f(\emptyset)$, and building up g to be defined on databases of increasing size. At each step, the algorithm ensures that no sensitivity constraints are violated and chooses the best value for $g(D)$ subject to those constraints.

For a given database D , $\text{UPPER}(D)$ is the maximum value $g(D)$ can take without letting it increase too much from a smaller database (violating an individual sensitivity parameter). Similarly, $\text{LOWER}(D)$ is the minimum value we can make $g(D)$ without letting it decrease too much from a smaller database. We then define $g(D)$ to be the value in $[\text{LOWER}(D), \text{UPPER}(D)]$ that is the closest to $f(D)$.

Algorithm 1 Sensitivity-Preprocessing Function Algorithm : $\text{PREPROCESSING}(f : \mathcal{D} \rightarrow \mathbb{R}, \{\Delta_i\}, D)$

Input: Function $f : \mathcal{D} \rightarrow \mathbb{R}$, individual sensitivity bounds $\{\Delta_i\}$, and database D of size n .

Output: $g(D)$, where g satisfies individual sensitivity Δ_i for all i .

Initialize $g(\emptyset) = f(\emptyset)$

for $k=1, \dots, n$ **do**

for every database $D' \subseteq D$ of size k **do**

 Set $\text{UPPER}(D') = \min_{i \in D'} \{g(D' - x_i) + \Delta_i\}$

 Set $\text{LOWER}(D') = \max_{i \in D'} \{g(D' - x_i) - \Delta_i\}$

 Set $g(D') = \begin{cases} \text{UPPER}(D'), & \text{if } \text{UPPER}(D') \leq f(D') \\ \text{LOWER}(D'), & \text{if } \text{LOWER}(D') \geq f(D') \\ f(D'), & \text{otherwise} \end{cases}$

end for

end for

Output $g(D)$

This construction of g ensures that the individual sensitivity of g does not exceed Δ_i for each i . We can then use these bounds on the sensitivity of g to calibrate the scale of noise that must be added to ensure differential privacy. In the special case that $\Delta_i = \Delta$ for all i , then the global sensitivity of g is Δ , and we can add noise that scales with $O(\frac{\Delta}{\epsilon})$ to achieve ϵ -differential privacy. Note that this guarantee holds even if f has unbounded sensitivity. In Section 7, we show how to satisfy differential privacy under heterogeneous Δ_i .

Remark 3.2. Our algorithm is initialized using $f(\emptyset)$, and thus centers g around this point. In the case that $f(\emptyset)$ is undefined—for example, when f computes the mean of a database—the analyst should initialize $g(\emptyset)$ using some domain knowledge or prior beliefs on reasonable centering of the function. If no prior knowledge is available, the analyst can sample multiple databases and evaluate f on the samples to estimate a reasonable centering point for $g(\emptyset)$. The sensitivity bounds will still hold regardless of the centering of g , but accuracy may suffer if $g(\emptyset)$ is set to be far from most values of f .

In addition to sensitivity guarantees and runtime analysis, we also provide an instance-specific error bound. Unfortunately this bound will not be in a clean form, but it does capture the intuitive fact that if we increase any Δ_i then it is likely that accuracy also increases.

However, we are able to obtain a bit more intuition on our instance-specific error bounds, and can consider them in a similar context to local sensitivity. Given that our Sensitivity-Preprocessing Function defines a database recursively in terms of its subsets, it makes sense that our error guarantees will be in terms of these subsets. These error bounds can then be seen as capturing the sensitivity between the neighboring subsets of D . Analogously to local sensitivity, we will have larger errors for databases with high sensitivity between the neighboring subsets.

Theorem 3.3. *Given $T(n)$ time query access to an arbitrary function $f : \mathcal{D} \rightarrow \mathbb{R}$, and sensitivity parameters $\{\Delta_i\}$, PREPROCESSING provides $O((T(n) + n)2^n)$ time access to the Sensitivity-Preprocessing Function $g : \mathcal{D} \rightarrow \mathbb{R}$ such that $\Delta_i(g) \leq \Delta_i$ for all i . Further, for any database $D = (x_1, \dots, x_n)$,*

$$|f(D) - g(D)| \leq \max_{\sigma \in \sigma_D} \sum_{i=1}^{|D|} \max\{|f(D_{\sigma(<i)} + x_{\sigma(i)}) - f(D_{\sigma(<i)})| - \Delta_{\sigma(i)}, 0\},$$

where σ_D is the set of all permutations on $[n]$, and $D_{\sigma(<i)} = (x_{\sigma(1)}, \dots, x_{\sigma(i-1)})$ is the subset of D that includes all individual data in the permutation before the i th entry.

Remark 3.4. We can easily extend this theorem to $f : \mathcal{D} \rightarrow \mathbb{R}^d$ by running PREPROCESSING on each dimension independently in terms of sensitivity parameters and error bounds. Specifically, suppose we were instead given parameters $\{\Delta_i\}$ where $\Delta_i = (\Delta_{i,1}, \dots, \Delta_{i,d})$ has different sensitivity parameters for each dimension of the function. We could then consider the function restricted to a single dimension d' , and run PREPROCESSING on this projection with sensitivity parameters $\{\Delta_{i,d'}\}$. This will give the desired sensitivity bounds in that single dimension, then running PREPROCESSING on all dimensions and composing across dimensions will give the appropriate Sensitivity-Preprocessing Function in d dimensions. In Section 8, we consider extensions to higher dimensions where each dimension is not treated independently.

3.2 Sensitivity-Preprocessing Function Correctness

We first prove that the Sensitivity-Preprocessing Function given in Definition 3.1 both meets the individual sensitivity criteria and is also defined on all databases.

Lemma 3.5. *For any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and non-negative sensitivity parameters $\{\Delta_i\}$, if $g : \mathcal{D} \rightarrow \mathbb{R}$ is defined according to Definition 3.1, then g is defined on all databases $D \in \mathcal{D}$ and $\Delta_i(g) \leq \Delta_i$ for all i .*

Proof. It suffices to show that for any $D \in \mathcal{D}$ with at least one entry and for any $x_i \in D$, we have $g(D - x_i) - \Delta_i \leq g(D) \leq g(D - x_i) + \Delta_i$. By our construction, this must always be true if $\text{LOWER}(D) \leq g(D) \leq \text{UPPER}(D)$ for any $D \in \mathcal{D} \setminus \{\emptyset\}$. Our construction of g will always place $g(D) \in [\text{LOWER}(D), \text{UPPER}(D)]$ if the interval is non-empty, so it suffices to show that for all $D \in \mathcal{D} \setminus \{\emptyset\}$,

$$\text{LOWER}(D) \leq \text{UPPER}(D).$$

We will prove this by induction starting with $D = x_i$ with one entry. Therefore, $\text{UPPER}(D) = f(\emptyset) + \Delta_i$ and $\text{LOWER}(D) = f(\emptyset) - \Delta_i$, which implies our desired inequality because $\Delta_i \geq 0$.

We now consider an arbitrary D and assume that our claim holds for all $D' \subset D$. Let $x_k \in D$ minimize $g(D - x_i) + \Delta_i$ over all $x_i \in D$, so

$$\text{UPPER}(D) = g(D - x_k) + \Delta_k,$$

and let $x_j \in D$ maximize $g(D - x_i) - \Delta_i$ over all $x_i \in D$, so

$$\text{LOWER}(D) = g(D - x_j) - \Delta_j.$$

If $k = j$ then the desired inequality immediately follows. Otherwise we consider $D - x_k - x_j$. By our inductive hypothesis, we know $\text{LOWER}(D - x_k) \leq g(D - x_k) \leq \text{UPPER}(D - x_k)$, so

$$g(D - x_k) \geq \text{LOWER}(D - x_k) \geq g(D - x_k - x_j) - \Delta_j.$$

Similarly, we have $\text{LOWER}(D - x_j) \leq g(D - x_j) \leq \text{UPPER}(D - x_j)$, so

$$g(D - x_j) \leq \text{UPPER}(D - x_j) \leq g(D - x_k - x_j) + \Delta_k.$$

Combining these inequalities gives $g(D - x_k) + \Delta_j \geq g(D - x_j) - \Delta_k$, which implies our desired result. \square

3.3 Error Bounds for Sensitivity-Preprocessing Function

We now prove the desired instance-specific error bounds between the original function and our Sensitivity-Preprocessing Function.

Lemma 3.6. *For any function $f : \mathcal{D} \rightarrow \mathbb{R}$ and non-negative sensitivity parameters $\{\Delta_i\}$, if $g : \mathcal{D} \rightarrow \mathbb{R}$ is defined according to Definition 3.1, then for any database $D \in \mathcal{D}$,*

$$|f(D) - g(D)| \leq \max_{\sigma \in \sigma_D} \sum_{i=1}^{|D|} \max\{|f(D_{\sigma(<i)} + x_{\sigma(i)}) - f(D_{\sigma(<i)})| - \Delta_{\sigma(i)}, 0\},$$

where σ_D is the set of all permutations on $[n]$, and $D_{\sigma(<i)} = (x_{\sigma(1)}, \dots, x_{\sigma(i-1)})$ is the subset of D that includes all individual data in the permutation before the i th entry.

Proof. We will prove this claim inductively and first consider $D = x_j$ with one entry for some j . We need to show

$$|f(D) - g(D)| \leq \max\{|f(D) - f(\emptyset)| - \Delta_j, 0\},$$

which follows easily from construction of g . We now consider an arbitrary D and assume that the claim is true for all $D' \subset D$. From our construction we claim that

$$|f(D) - g(D)| \leq \max_{x_i \in D} \{|f(D) - g(D - x_i)| - \Delta_i, 0\}.$$

This follows from the fact that if $f(D) = g(D)$ then we must have $|f(D) - g(D - x_i)| \leq \Delta_i$ for all i , and otherwise there must be some $x_i \in D$ such that the constraint on $g(D)$ with respect to Δ_i is tight. Using this fact we can bound $|f(D) - g(D)|$ in the following way:

$$\begin{aligned} |f(D) - g(D)| &\leq \max_{x_i \in D} \{|f(D) - g(D - x_i)| - \Delta_i, 0\} \\ &= \max_{x_i \in D} \{|f(D) - f(D - x_i) + f(D - x_i) - g(D - x_i)| - \Delta_i, 0\} \\ &\leq \max_{x_i \in D} \{|f(D) - f(D - x_i)| - \Delta_i + |f(D - x_i) - g(D - x_i)|, 0\} \\ &\leq \max_{x_i \in D} \{\max\{|f(D) - f(D - x_i)| - \Delta_i, 0\} + |f(D - x_i) - g(D - x_i)|\} \end{aligned}$$

We then apply the inductive hypothesis to $|f(D - x_i) - g(D - x_i)|$, which immediately implies our desired bound. \square

3.4 Proof of Theorem 3.3

Proof of Theorem 3.3. The individual sensitivity guarantees are given by Lemma 3.5, and the error bounds are given by Lemma 3.6. It then remains to show the running time. If we assume $T(n)$ time access to f for a database with n entries, then because we need to query each subset of D , this will contribute time $O(T(n)2^n)$. Furthermore, for each subset we need to compute $\text{UPPER}(D)$ and $\text{LOWER}(D)$ which takes $O(n)$ time for each subset. This then gives our full runtime of $O((T(n) + n)2^n)$. \square

4 Optimality and Hardness of Sensitivity-Preprocessing Function

Our algorithm in Section 3 took exponential time to query the Sensitivity-Preprocessing Function g at each database D of interest, and, while we did achieve bounds on the error incurred, their complicated formulation makes it difficult to determine whether these bounds are strong. In this section we give strong justification for our construction of the Sensitivity-Preprocessing Function in terms of both error incurred and the exponential running time for the general setting.

In Section 4.1 we consider the general problem of approximating an arbitrary function $f : \mathcal{D} \rightarrow \mathbb{R}$ with one that has individual sensitivity bounded by $\{\Delta_i\}$. Under the ℓ_∞ metric, our Sensitivity-Preprocessing Function will achieve a 2-approximation of the optimal function. Furthermore, this 2-approximation can still be obtained when the optimal function is restricted to certain subsets of the data universe. Informally, this will imply that on subsets which allow for small error between f and a function with individual sensitivity bounded by $\{\Delta_i\}$, our Sensitivity-Preprocessing Function will also have small error. Due to ℓ_∞ being a worst-case metric, it is then natural to ask if our Sensitivity-Preprocessing Function actually still performs well on the non-worst-case databases. To this end, we show that our Sensitivity-Preprocessing Function is Pareto optimal, meaning that for any other function with individual sensitivity bounded by $\{\Delta_i\}$, if it has smaller error on some database relative to our Sensitivity-Preprocessing Function, then there must exist another database on which it has higher error.

In Section 4.2 we show that it is NP-hard to achieve our approximation guarantees with respect to the ℓ_∞ metric. We further show that it is uncomputable to do better than a 2-approximation in the ℓ_∞ metric, and also uncomputable to achieve even a constant approximation in any ℓ_p metric for $p < \infty$ which justifies our choice of metric. We believe that the combination of these results gives a strong indication that our Sensitivity-Preprocessing Function and corresponding exponential time construction is the best we can hope to achieve for the general problem.

4.1 Optimality guarantees

In this section we prove that our Sensitivity-Preprocessing Function achieves certain optimality guarantees. As there are many ways in which to measure how close one function is to another, it is first necessary to be more specific about the definition of optimality we use here. The set that we are trying to optimize over will be all functions with bounded individual sensitivity:

Definition 4.1. Given a data universe \mathcal{D} and individual sensitivity parameters $\{\Delta_i\}$, define

$$F_{\{\Delta_i\}}(\mathcal{D}) \stackrel{\text{def}}{=} \{f : \mathcal{D} \rightarrow \mathbb{R} \mid \Delta_i(f) \leq \Delta_i, \forall i\}.$$

In this context, the general goal will then be to show that our Sensitivity-Preprocessing Function is close to the optimal function on this set. Here we will consider optimal to be under the ℓ_∞ metric, where we want $f^* \in F_{\{\Delta_i\}}(\mathcal{D})$ to minimize the maximum difference $|f(D) - f^*(D)|$ over all $D \in \mathcal{D}$. Our Sensitivity-Preprocessing Function achieves a 2-approximation to the optimal $f^* \in F_{\{\Delta_i\}}(\mathcal{D})$ with respect to the ℓ_∞ metric. For unbounded sensitivity functions, the value $|f(D) - f^*(D)|$ will be unbounded, so we will instead show the stronger result that this 2-approximation also holds if we restrict the data universe to a single database and its subsets. Specifically, we show that if for certain subsets of the data universe it is possible to perfectly fit f to a $\{\Delta_i\}$ individual sensitivity bounded function, then our Sensitivity-Preprocessing Function will also perfectly fit to f in this subset. These guarantees are formalized in the following lemma.

Lemma 4.2. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with parameters $\{\Delta_i\}$. For any arbitrary $D \in \mathcal{D}$, define $\mathcal{D}' = \{D' \subseteq D\}$. Then,*

$$\max_{D' \in \mathcal{D}'} |f(D') - g(D')| \leq 2 \min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D}')} \max_{D' \in \mathcal{D}'} |f(D') - f^*(D')|$$

Proof. We will prove this inductively on the size of D . It is immediately true for $D = \emptyset$. We now prove for arbitrary D where we assume the claim for all strict subsets of D . Our proof will be by contradiction, where we suppose that our claim is not true for some D .

We first determine the database at which $|f(D') - g(D')|$ is maximized. Suppose $\arg \max_{D' \in \mathcal{D}'} |f(D') - g(D')| = \tilde{D}$ such that $\tilde{D} \subset D$. Define $\tilde{\mathcal{D}} = \{D' \subseteq \tilde{D}\}$. Because $\tilde{D} \subset D$, it must follow that

$$\min_{f^* \in F_{\{\Delta_i\}}(\tilde{\mathcal{D}})} \max_{D' \in \tilde{\mathcal{D}}} |f(D') - f^*(D')| \leq \min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D}')} \max_{D' \in \mathcal{D}'} |f(D') - f^*(D')|.$$

By our assumption that the claim is not true on D , it follows that

$$|f(\tilde{D}) - g(\tilde{D})| > 2 \min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D}')} \max_{D' \in \mathcal{D}'} |f(D') - f^*(D')|.$$

Combining this with the previous inequality implies,

$$|f(\tilde{D}) - g(\tilde{D})| > 2 \min_{f^* \in F_{\{\Delta_i\}}(\tilde{\mathcal{D}})} \max_{D' \in \tilde{\mathcal{D}}} |f(D') - f^*(D')|,$$

which contradicts our inductive hypothesis. Therefore we must have $\max_{D' \in \mathcal{D}'} |f(D') - g(D')| = |f(D) - g(D)|$.

We now apply Lemma 4.3, which we prove subsequently, to see that there must exist $\tilde{D} \subset D$ such that $|f(D) - f(\tilde{D})| \geq |f(D) - g(D)| + \sum_{i \in D \setminus \tilde{D}} \Delta_i$. Therefore for any $f^* \in F_{\{\Delta_i\}}(\mathcal{D}')$ it must be true that

$$\max\{|f(\tilde{D}) - f^*(\tilde{D})|, |f(D) - f^*(D)|\} \geq \frac{|f(D) - g(D)|}{2},$$

because of the sensitivity constraints. We then use the fact that $\max_{D' \in \mathcal{D}'} |f(D') - g(D')| = |f(D) - g(D)|$ to conclude,

$$\max_{D' \in \mathcal{D}'} |f(D') - g(D')| \leq 2 \min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D}')} \max_{D' \in \mathcal{D}'} |f(D') - f^*(D')|.$$

This contradicts our assumption, so the claim must therefore be true for D . \square

Lemma 4.3. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with individual sensitivity parameters $\{\Delta_i\}$. For any $D \in \mathcal{D}$ such that $f(D) \neq g(D)$ there must exist some $\tilde{D} \subset D$ such that $g(D) \geq f(\tilde{D}) + \sum_{i \in D \setminus \tilde{D}} \Delta_i$ if $f(D) > g(D)$ and $g(D) \leq f(\tilde{D}) - \sum_{i \in D \setminus \tilde{D}} \Delta_i$ if $f(D) < g(D)$.*

Proof. We prove the claim inductively, starting with the immediate observation that by construction it is true when D only has one entry.

We now consider an arbitrary D and assume our claim for all subsets. Without loss of generality, we will prove the claim if $f(D) > g(D)$, and can symmetrically apply the proof for the case when $f(D) < g(D)$. If $f(D) > g(D)$, then there must exist some $x_i \in D$ such that $g(D) = g(D - x_i) + \Delta_i$. If $f(D - x_i) \leq g(D - x_i)$, then we can set $\tilde{D} = D - x_i$ and the claim follows. Otherwise we must have $f(D - x_i) > g(D - x_i)$ and we apply our inductive hypothesis to obtain some $\tilde{D} \subset D - x_i$ such that

$$g(D - x_i) \geq f(\tilde{D}) + \sum_{j \in (D - x_i) \setminus \tilde{D}} \Delta_j.$$

We then use the fact that $g(D) = g(D - x_i) + \Delta_i$ to achieve

$$g(D) \geq f(\tilde{D}) + \sum_{j \in D \setminus \tilde{D}} \Delta_j.$$

□

We note that because Lemma 4.2 achieves a 2-approximation when the optimal function is restricted to subsets of the data universe, we easily achieve a 2-approximation on the full data universe.

Corollary 4.4. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with parameters $\{\Delta_i\}$. Then,*

$$\max_{D' \in \mathcal{D}} |f(D') - g(D')| \leq 2 \min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D})} \max_{D' \in \mathcal{D}} |f(D') - f^*(D')|.$$

Pareto Optimality

We now complement our localized 2-approximation of the ℓ_∞ metric with a Pareto optimality result. As ℓ_∞ is a worst-case metric we would still like our Sensitivity-Preprocessing Function to perform well on the non-worst-case databases. In particular, for the databases that do not contribute to the ℓ_∞ error, we still want the error to be minimized. The following lemma will conclude that we cannot improve the error of a single database without incurring more error on another database, indicating that we are still performing well on the non-worst-case databases.

Lemma 4.5. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with individual sensitivity parameters $\{\Delta_i\}$. For any $h \in F_{\{\Delta_i\}}(\mathcal{D})$ if there is some $D \in \mathcal{D}$ such that*

$$|f(D) - h(D)| < |f(D) - g(D)|,$$

then there also exists some $D' \in \mathcal{D}$ such that

$$|f(D') - h(D')| > |f(D') - g(D')|.$$

Proof. Suppose there is some $h \in F_{\{\Delta_i\}}(\mathcal{D})$ such that

$$|f(D) - h(D)| < |f(D) - g(D)|$$

for some $D \in \mathcal{D}$, and for all $D' \in \mathcal{D}$,

$$|f(D') - h(D')| \leq |f(D') - g(D')|$$

Then it must be true that $h(\emptyset) = g(\emptyset)$ because $g(\emptyset) = f(\emptyset)$. Let D be the smallest database such that $h(D) \neq g(D)$, which implies that $|f(D) - h(D)| < |f(D) - g(D)|$. This inequality implies $g(D) \neq f(D)$, and by our construction of g , either $\text{UPPER}(D) < f(D)$ or $\text{LOWER}(D) > f(D)$.

Without loss of generality, assume $\text{UPPER}(D) < f(D)$ and thus $g(D) = \text{UPPER}(D)$. Using the fact that $|f(D) - h(D)| < |f(D) - g(D)|$, we can conclude that $h(D) > g(D)$. However, since $\text{UPPER}(D) = g(D - x_i) + \Delta_i$ for some $x_i \in D$, we must have $h(D) > g(D - x_i) + \Delta_i$. Our assumption that D was the smallest database such that $h(D) \neq g(D)$ then implies $h(D) > h(D - x_i) + \Delta_i$, contradicting the individual sensitivity of i being at most Δ_i in h .

Therefore, $F_{\{\Delta_i\}}(\mathcal{D})$ cannot contain such an h , which implies our claim. \square

4.2 Hardness of approximation

In this section we justify the exponential running time of our implementation of the Sensitivity-Preprocessing Function for the general setting. Recall that in our construction we did not make any assumptions about \mathcal{D} and only required query access to the function $f : \mathcal{D} \rightarrow \mathbb{R}$. Under this limited knowledge setting it is reasonable that our localized greedy construction is the best we can hope for, despite taking exponential time. Accordingly, we show here that even if we restrict \mathcal{D} to be exponential-sized, set all $\{\Delta_i\}$ to be the same Δ , and further force f to be polytime representable, it is still NP-hard to compute our Sensitivity-Preprocessing Function. This proof will further imply that it is NP-hard to compute a function that has identical individual sensitivity guarantees and achieve the same approximation guarantees that our Sensitivity-Preprocessing Function does in Lemma 4.2.

After proving this NP-hardness result, we will discuss the issues with computing individual sensitivity bounded functions that obtain better approximations. We give strong justification that it is uncomputable to achieve better than a 2-approximation in the ℓ_∞ metric. Further, we give similar reasons why it is uncomputable to achieve even a constant approximation on average error for the general setting, which justifies our choice of metric for proving our approximation guarantees in the previous section. We believe these ideas could be formalized in a straightforward manner, but think that doing so is unnecessary for the scope of this paper.

NP-hardness

Proposition 4.6. *For certain $f : \mathcal{D} \rightarrow \mathbb{R}$ such that $|\mathcal{D}| = O(3^n)$, it is NP-hard to compute our Sensitivity-Preprocessing Function g with parameter Δ on a specific database.*

Proof. In order to prove this claim, we will construct a gadget function that takes an arbitrary SAT formula ϕ and constructs a function $f : \mathcal{D} \rightarrow \mathbb{R}$ such that $|\mathcal{D}| = O(3^n)$ and on a specified database $D \in \mathcal{D}$, $g(D) < n$ if and only if ϕ is satisfiable. We construct that gadget function below.

Gadget Function: Let \mathcal{D} be the data universe with n individuals such that $x_i \in \{T, F\}$. Let $\phi : \{T, F\}^n \rightarrow \{0, 1\}$ be an arbitrary SAT formula of n variables that outputs 0 if false and 1 if true. For any $D \in \mathcal{D}$, let $D + T \in \{T, F\}^n$ be the assignment of variables that correspond to D and set all variables not in D to be true. Let the function $f_\phi : \mathcal{D} \rightarrow \mathbb{R}$ be defined as $f_\phi(D) = |D| - \phi(D + T)$ where $|D| = |\{i \in D\}|$. Further, define $f_\phi(\emptyset) = 0$ and let $\Delta = 1$.

Claim: For the constructed gadget function f from SAT formula ϕ and our corresponding Sensitivity-Preprocessing Function g with parameter Δ , we must have that $g(F^n) < n$ iff ϕ is satisfiable.

First, we assume ϕ is unsatisfiable which implies $f_\phi(D) = |D|$ for all D . Therefore the sensitivity of f is 1, and g will be identical to f , so $g(F^n) = n$.

Next, we show that if $g(F^n) \geq n$ then there cannot exist a satisfying assignment of ϕ . Suppose there does exist a satisfying assignment, then take the one with the fewest false assignments and denote this as $x^* \in \{T, F\}^n$. Further, consider the database $D \subseteq F^n$ that consists of all of the false assignments of x^* . By definition, we must have that $f_\phi(D) = |D| - 1$, and we further show that $g(D) = |D| - 1$.

For any $D' \subset D$, we have $f_\phi(D') = |D'|$ by construction of f_ϕ and our assumption that x^* was the satisfying assignment with the fewest false assignments. It is easy to see that $g(D') = |D'|$ by construction, which implies that $g(D) = |D| - 1$. Since the sensitivity is set to be 1, we have that for every \tilde{D} such that $\tilde{D} \supseteq D$ it must be true that $g(\tilde{D}) \leq |\tilde{D}| - 1$. By construction, we know $D \subseteq F^n$, which implies $g(F^n) < n$. This gives a contradiction and implies that ϕ is unsatisfiable. \square

Note that to satisfy the approximation guarantees given in Lemma 4.2, any $f^* \in F_\Delta(\mathcal{D})$ would require $f^*(D) = |D| - 1$ in our proof as well. Accordingly, for any $f^* \in F_\Delta(\mathcal{D})$ that satisfies the approximation guarantees of Lemma 4.2, it must also be true that $f(F^n) < n$ iff ϕ is satisfiable. Therefore, any algorithm that achieves the same guarantees must also be NP-hard to compute.

Uncomputability of better approximations

We now argue that it is uncomputable to achieve better approximation factors than our Sensitivity-Preprocessing Function, with respect to both the ℓ_∞ metric and any ℓ_p metric.

Remark 4.7. We claim that no finitely computable algorithm can obtain a function with appropriately bounded individual sensitivities that achieves better than a 2-approximation on the ℓ_∞ error. Let \mathcal{D} only contain the empty set and databases of size one, each containing a single real-valued data entry $x \in [0, 1]$, and set $\Delta = 1$. Consider any finite algorithm that constructs a Δ -sensitivity function h to minimize the maximum difference between (adversarially chosen) f and h over all databases.

If f is arbitrary and only query accessible, then the algorithm can only query a finite number of databases, and an adversary could just set $f(x) = f(\emptyset) = 0$ for all queried databases. In order to achieve even a constant approximation, the algorithm would need to set $f(x) = 0$ just in case $f(x) = 0$ for all $x \in [0, 1]$. However, the adversary could then set $f(y) = 2$ for all non-queried databases. The function that minimizes the ℓ_∞ error would then set $f(x) = 1/2$ for all queried databases and $f(y) = 3/2$ for all non-queried databases. As a result, the finite algorithm can only achieve a 2-approximation.

Remark 4.8. We further claim that no finitely computable algorithm can obtain a function with appropriately bounded individual sensitivities that achieves a constant approximation on the average ℓ_p error. The optimal function in this scenario would be f^* that minimizes:

$$\min_{f^* \in F_{\{\Delta_i\}}(\mathcal{D})} \left(\frac{\sum_{D \in \mathcal{D}} (f(D) - f^*(D))^p}{|\mathcal{D}|} \right)^{1/p}.$$

We consider the same example as above, and note that the number of queried databases is finite and the number of non-queried databases is infinite. In order to achieve a constant approximation, the algorithm would need to set $f(x) = 0$ just in case $f(x) = 0$ for all $x \in [0, 1]$. However, it would then have to set $f(y) = 1$ for all non-queried databases and the average ℓ_p error would be a constant.

If instead it set $f(x) = 1$ for all queried databases and $f(y) = 2$ for all non-queried databases, then the average ℓ_p error would approach 0 because the non-queried databases are infinite and the queried databases are finite. As a result, no finitely computable algorithm can achieve a constant approximation in this metric.

5 Efficient Implementation of Several Statistical Measures

In this section, we take our general recursive algorithm and show how it can be made efficient for a variety of important statistical measures such as mean, α -trimmed mean, median, minimum, and maximum. It is important to note that we will not change the key recursive structure, but instead show that when we have more information about the function, we can ignore many of the subproblems of the recursion for significant runtime speedups. As a result, the algorithm given for these statistical tasks will take $O(n^2)$ time and have a simple dynamic programming construction.

The key idea will be that given a database $D = (x_1, \dots, x_n)$ where we assume for simplicity that $x_1 \leq \dots \leq x_n$,⁵ the only important subproblems will be $D - x_1$ and $D - x_n$. Consequently, instead of considering every possible subset of D , we only need to consider every contiguous subset, which limits the number of subproblems to $O(n^2)$.

We first give a general class of functions—which includes mean, median, α -trimmed mean, minimum, and maximum—for which it is straightforward to show our algorithm can be applied efficiently. We then give a more in-depth analysis of the error guarantees that correspond with this implementation for mean. These bounds will ultimately be quite intuitive, but the proofs will be more involved.

5.1 Efficient implementation for a simple class of functions

We will first define a class of functions under which database ordering is preserved for any subset, which allows us to presort the data according to this ordering and restrict the number of subproblems. Intuitively, it implies that for any database $D = (x_1, \dots, x_n)$ there is an ordering of the x_1, \dots, x_n such that the extreme points in our recursion are determined by the databases that remove the maximum or the minimum. In particular, if we consider the mean function $\mu : \mathbb{R}^{\leq n} \rightarrow \mathbb{R}$ then for any $D = (x_1, \dots, x_n)$ if we assume $x_1 \leq \dots \leq x_n$, then we know $\mu(D - x_n) \leq \mu(D - x_i)$ and $\mu(D - x_1) \geq \mu(D - x_i)$ for any i . This will ultimately imply that our upper and lower bounds on the allowable region for $g(D)$ will be defined by $g(D - x_1)$ and $g(D - x_n)$, respectively.

Definition 5.1 (Database-ordered function). A function $f : \mathcal{D} \rightarrow \mathbb{R}$ is *database-ordered* if for any $D = (x_1, \dots, x_n) \in \mathcal{D}$ and any pair $x_i, x_j \in D$, we have that for every subset database $D' \subset D$ such that $x_i, x_j \notin D'$, then either $f(D' + x_i) \leq f(D' + x_j)$ for every D' or $f(D' + x_i) \geq f(D' + x_j)$ for every D' . Furthermore, if $f(D' + x_i) \leq f(D' + x_j)$ for every D' , we say that $x_i \leq x_j$ in the *entry-ordering*, and vice-versa if $f(D' + x_i) \geq f(D' + x_j)$ for every D' .

The general idea of our efficient implementation will be to use the ordering and only consider contiguous subsets according to this ordering.

Lemma 5.2. *Given a database-ordered function $f : \mathcal{D} \rightarrow \mathbb{R}$, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of f with parameter Δ . Then for any $D = (x_1, \dots, x_n)$ where $x_1 \leq \dots \leq x_n$ in the entry-ordering we must have $\text{UPPER}(D) = g(D - x_n) + \Delta$ and $\text{LOWER}(D) = g(D - x_1) - \Delta$, and our PREPROCESSING algorithm only requires solving $O(n^2)$ subproblems*

⁵Our algorithm will presort and only incur $O(n \log n)$ running time.

Proof. We first want to show $\text{UPPER}(D) = g(D - x_n) + \Delta$ and $\text{LOWER}(D) = g(D - x_1) - \Delta$. It is sufficient to show $g(D - x_n) \leq g(D - x_{n-1}) \leq \dots \leq g(D - x_1)$, which we will prove by induction on the size of the database. If D only has one entry, then this must be true.

Assume this is true for all D with at most $n - 1$ entries, and we want to show $g(D - x_{i+1}) \leq g(D - x_i)$ for any $i \in [n - 1]$. Since f is database-ordered, we know that $f(D - x_{i+1}) \leq f(D - x_i)$. It then suffices to show $\text{UPPER}(D - x_{i+1}) \leq \text{UPPER}(D - x_i)$ and $\text{LOWER}(D - x_{i+1}) \leq \text{LOWER}(D - x_i)$. By our inductive hypothesis, $\text{UPPER}(D - x_i) = g(D - x_i - x_n) + \Delta$ and $\text{UPPER}(D - x_{i+1}) = g(D - x_{i+1} - x_n) + \Delta$ if $i < n - 1$, and we note that $\text{UPPER}(D - x_{n-1}) = \text{UPPER}(D - x_n)$. Also by our inductive hypothesis, $g(D - x_{i+1} - x_n) \leq g(D - x_i - x_n)$, implying $\text{UPPER}(D - x_{i+1}) \leq \text{UPPER}(D - x_i)$. The proof for $\text{LOWER}(D - x_{i+1}) \leq \text{LOWER}(D - x_i)$ follows symmetrically.

With this fact, it is straightforward to see that opening up our algorithm, instead of considering all subsets of size k , it suffices to consider subsets $(x_1, \dots, x_k), (x_2, \dots, x_{k+1}), \dots, (x_{n-k}, \dots, x_n)$. Then the total number of subproblems that need to be solved is $O(n^2)$. \square

If our function is efficiently computable and the entry-ordering is efficiently computable, this then gives an efficient implementation of our recursive algorithm. In particular, for several functions of statistical interest including mean, α -trimmed mean, median, maximum, and minimum, this easily yields an efficient algorithm.

Algorithm 2 Efficient Implementation for database-ordered functions

Input: Database-ordered function $f : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$, sensitivity bound Δ , estimate for the empty set $\hat{\mu}$, and database $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ for some arbitrary n .

Output: $g(D)$, where g is the Sensitivity-Preprocessing Function of f .

Initialize $g(\emptyset) = \hat{\mu}$

Sort D (We will assume $x_1 \leq \dots \leq x_n$ for simplicity)

for $k=1, \dots, n$ **do**

for $i = 1, \dots, n-k+1$ **do**

for every database $D' = (x_i, \dots, x_{i+k-1})$ **do**

 Let $g(D') = \begin{cases} g(D' - x_{i+k-1}) + \Delta, & \text{if } g(D' - x_{i+k-1}) + \Delta \leq f(D') \\ g(D' - x_i) - \Delta, & \text{if } g(D' - x_i) - \Delta \geq f(D') \\ f(D'), & \text{otherwise} \end{cases}$

end for

end for

end for

Output $g(D)$

Corollary 5.3. *We can implement our Sensitivity-Preprocessing Function with parameter Δ in $O(n^2)$ time for the functions mean, α -trimmed mean, median, maximum, and minimum.*

Proof. Let f be any of the functions listed above. It is simple to see that for any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$, and any $y, z \in \mathbb{R}$, if $y \leq z$ then $f(D+y) \leq f(D+z)$, and if $y \geq z$ then $f(D+y) \geq f(D+z)$. This implies that f is database-ordered, then by Lemma 5.2 we only need to solve $O(n^2)$ subproblems.

Further, we note that finding the entry-ordering simply requires sorting the entries of D in $O(n \log n)$ time. If the database is ordered, then computing median, minimum, and maximum only requires $O(1)$ time. If we know the mean or α -trimmed mean for $D - x_i$ for some x_i , we can compute the mean or α -trimmed mean of D in $O(1)$ time using the fact that

$$\frac{x_1 + \dots + x_n}{n} = \frac{n-1}{n} \left(\frac{x_1 + \dots + x_{n-1}}{n-1} \right) + \frac{x_n}{n}$$

Note that we compute $D - x_i$ for some i in our subproblems, so we will in fact have access to this value. As a result, the full running time will take $O(n^2)$ time. \square

5.2 Improved runtime and accuracy for median

In the previous section, we showed that for several important statistical measures we could give a simple efficient version of our general algorithm. To complement this result, we further examine the median function and give an improved analysis that requires only $O(n)$ time for presorted data and provides strong accuracy guarantees. Improving the running time will utilize the critical property that removing the minimum and maximum value does not change the median. As was seen in our previous section, our recursion was reduced by only considering removing the maximum or minimum value. The related fact regarding median will be incorporated into an inductive claim that we never overshoot the true median, and can further reduce our recursion.

Lemma 5.4. *Let $med : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the median function and $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the Sensitivity-Preprocessing Function of med with parameter Δ . Then for any $D = (x_1, \dots, x_n)$ such that $x_1 \leq \dots \leq x_n$, computing $g(D)$ takes $O(n)$ time.*

Proof. It follows immediately from Lemma 5.2 and Lemma 5.5 that if $med(D) \geq med(\emptyset)$ then $g(D) = \min\{med(D), g(D - x_n) + \Delta\}$ and otherwise $g(D) = \max\{med(D), g(D - x_1) - \Delta\}$. We can calculate $med(D)$ and any contiguous subset of D in $O(1)$ time, and the recursion will only be upon one subproblem, implying a runtime of $O(n)$. \square

Lemma 5.5. *If $med(D) \geq med(\emptyset)$, then $med(\emptyset) \leq g(D) \leq med(D)$*

Proof. The proof will be inductive, and it is easy to verify that the inequality holds for $|D| \leq 2$. We then consider an arbitrary $D = (x_1, \dots, x_n)$ where we assume without loss of generality that $x_1 \leq \dots \leq x_n$ and $n \geq 3$. The critical fact we use here will be that the median does not change if you remove the minimum and maximum values, which is to say that $med(D) = med(D - x_1 - x_n)$. Therefore, if $med(D) \geq med(\emptyset)$, then we must also have $med(D - x_1 - x_n) \geq med(\emptyset)$, which by our inductive claim implies that $med(\emptyset) \leq g(D - x_1 - x_n) \leq med(D - x_1 - x_n) = med(D)$. Applying Lemma 5.2, we then have

$$g(D - x_1) \leq g(D - x_1 - x_n) + \Delta \leq med(D) + \Delta$$

and

$$g(D - x_n) \geq g(D - x_1 - x_n) - \Delta \geq med(D) - \Delta$$

We then reapply Lemma 5.2 to achieve our desired result that $med(\emptyset) \leq g(D) \leq med(D)$ \square

As in [NRS07], define

$$A^{(k)}(D) = \max_{d(D, D') \leq k} LS_f(D').$$

which is the k -local sensitivity of function f for database D . For odd n , this just reduces to $A^{(k)}(D) = \max_{0 \leq t \leq k+1} (x_{m+t} - x_{m+t-k-1})$ and $m = \frac{n+1}{2}$. It is similar for n is even, and essentially bounds the distance of each value from the median.

Combining this assumption with our previous lemma will then allow for stronger bounds upon $g(D)$.

Lemma 5.6. *Given some parameter Δ and $med(\emptyset)$, if $A^{(k)}(D) \leq 2(k+1)\Delta$ for $k \leq n/4$ and $med(D) \in [med(\emptyset) - \frac{n}{2}\Delta, med(\emptyset) + \frac{n}{2}\Delta]$, then $g(D) = med(D)$*

Proof. Without loss of generality, assume that $\text{med}(D) \geq \text{med}(\emptyset)$. By Lemma 5.5 we know $g(D) \leq \text{med}(D)$, then applying Lemma 5.7 gives our desired result. \square

Lemma 5.7. *Given some parameter Δ and $\text{med}(\emptyset)$, assume $A^{(k)}(D) \leq 2(k+1)\Delta$ for $k \leq n/4$ and $\text{med}(D) \in [\text{med}(\emptyset) - \frac{n}{2}\Delta, \text{med}(\emptyset) + \frac{n}{2}\Delta]$. Let $D_{[1:k]} = (x_1, \dots, x_k)$, if $\text{med}(D) \geq \text{med}(\emptyset)$, then $g(D_{[1:k]}) \geq \text{med}(D) - (n-k)\Delta$*

Proof. It is straightforward to see that our assumptions imply

$$\text{med}(D_{[1:k]}) \geq \text{med}(D) - (n-k)\Delta$$

for any $k \geq n/2$. We then consider our base case to be $k = n/2$, and note that from Lemma 5.5 we have $g(D_{[1:k]}) \geq \min\{\text{med}(\emptyset), \text{med}(D_{[1:k]})\}$, which by our assumptions immediately implies $g(D_{[1:n/2]}) \geq \text{med}(D) - \frac{n}{2}\Delta$.

We then assume this is true for $k-1 \geq n/2$, so $g(D_{[1:k-1]}) \geq \text{med}(D) - (n-k)\Delta - \Delta$. We also know from Lemma 5.2 that

$$g(D_{[1:k]}) \geq \min\{\text{med}(D_{[1:k]}), g(D_{[1:k-1]}) + \Delta\}$$

which implies our desired inequality. \square

5.2.1 Proof of Theorem 1.6

We now have all the necessary components to give our proof of Theorem 1.6, which we restate and prove below.

Theorem 1.6. *Let $\text{med} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the median function for the data universe of all finite-length real-valued vectors. For chosen parameters $\text{med}(\emptyset)$ and Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, if $x_1 \leq \dots \leq x_n$ we give $O(n)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that $g(D) = \text{med}(D)$ whenever $A^{(k)}(D) \leq 2(k+1)\Delta$ for $k \leq n/4$ and $\text{med}(D) \in [\text{med}(\emptyset) - \frac{n}{2}\Delta, \text{med}(\emptyset) + \frac{n}{2}\Delta]$.*

Proof of Theorem 1.6. The runtime guarantees follow immediately from Lemma 5.4. Furthermore, if we assume that $\text{med}(D) \geq \text{med}(\emptyset)$, then Lemma 5.5 implies that $g(D) \leq \text{med}(D)$ and Lemma 5.7 implies that $g(D) \geq \text{med}(D)$ because we have the same assumptions, and so $g(D) = \text{med}(D)$. The symmetric version of these lemmas follows immediately, and we also have $g(D) = \text{med}(D)$ when $\text{med}(D) \leq \text{med}(\emptyset)$. \square

5.3 Accuracy bounds for mean

We next consider the mean function, and provide strong bounds on the accuracy of our Sensitivity-Preprocessing Function. While the analysis will be rather involved, we believe that the ultimate guarantees are highly intuitive. Our proof will also show that for databases with entries bounded in a Δ sensitivity range, we perfectly preserve the accuracy between our new function and the mean function. Further, the key ideas in our proof are closely related to the construction of our recursive function, and we believe could be extended to other functions using a similar framework.

The general proof idea will be to give two simpler recursive functions that yield reasonably tight upper and lower bounds on our function. Due to their further simplicity, it will be much easier to give nice error bounds with respect to the true mean for these functions.

The idea behind constructing the upper and lower bound functions will be simple. Recall that we showed our g for the mean function has the property that $\text{UPPER}(D) = g(D - x_n) + \Delta$ and $\text{LOWER}(D) = g(D - x_1) - \Delta$ because we showed $g(D - x_n) \leq g(D - x_{n-1}) \leq \dots \leq g(D - x_1)$ if we assume $x_1 \leq \dots \leq x_n$. Intuitively, this is due to the fact that removing the maximum value will minimize mean and removing the minimum value will maximize mean. Accordingly, we will just iteratively remove the maximum value to give a lower bound on our function and iteratively remove the minimum value to give an upper bound on our function. These functions then only require solving $O(n)$ subproblems which will simplify the analysis.

Definition 5.8 (Mean-bounding functions). For any $D = (x_1, \dots, x_n)$, define

$$h_{\text{lower}}(D) = \begin{cases} h_{\text{lower}}(D - x_n) + \Delta, & \text{if } h_{\text{lower}}(D - x_n) + \Delta \leq \mu(D) \\ h_{\text{lower}}(D - x_n) - \Delta, & \text{if } h_{\text{lower}}(D - x_n) - \Delta \geq \mu(D) \\ \mu(D), & \text{otherwise} \end{cases}$$

and

$$h_{\text{upper}}(D) = \begin{cases} h_{\text{upper}}(D - x_1) + \Delta, & \text{if } h_{\text{upper}}(D - x_1) + \Delta \leq \mu(D) \\ h_{\text{upper}}(D - x_1) - \Delta, & \text{if } h_{\text{upper}}(D - x_1) - \Delta \geq \mu(D) \\ \mu(D), & \text{otherwise} \end{cases}$$

We will first show that h_{upper} and h_{lower} are upper and lower bounds, respectively, of our Sensitivity-Preprocessing Function g with parameter Δ . Then we further examine the properties of these functions.

Lemma 5.9. Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function with chosen parameters $\hat{\mu}$ and Δ . For any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ with $x_1 \leq x_2 \leq \dots \leq x_n$, then $h_{\text{lower}}(D) \leq g(D)$ and $h_{\text{upper}}(D) \geq g(D)$ where $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ is our Sensitivity-Preprocessing Function with parameter Δ .

Proof. We will prove both inequalities by induction, where we first note that if D only has one entry, then by construction $h_{\text{lower}}(D) = g(D) = h_{\text{upper}}(D)$.

For any database D of n entries, by induction we have $h_{\text{lower}}(D - x_n) \leq g(D - x_n)$ and note that within the proof of Lemma 5.2 we showed $g(D - x_n) \leq g(D - x_1)$, which implies $h_{\text{lower}}(D) \leq g(D)$. Similarly, by induction we have $h_{\text{upper}}(D - x_1) \geq g(D - x_1)$ and Lemma 5.2 gives $g(D - x_1) \geq g(D - x_n)$, which implies $h_{\text{upper}}(D) \leq g(D)$. \square

We now use the simpler recursive structure of h_{lower} and h_{upper} to get more explicit forms of their output.

Lemma 5.10. Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function with chosen parameters $\hat{\mu}$ and Δ . For any $D = (x_1, \dots, x_n)$, assume that $x_1 \leq x_2 \leq \dots \leq x_n$, and let $D_{[i:j]} = (x_i, \dots, x_j)$. Let k be the largest index such that $h_{\text{lower}}(D_{[1:k]}) \geq \mu(D_{[1:k]})$ (if one exists), then

$$h_{\text{lower}}(D_{[1:k]}) = \max\{\hat{\mu} - k\Delta, \mu(D_{[1:k]})\}.$$

Let l be the smallest index such that $h_{\text{upper}}(D_{[l:n]}) \geq \mu(D_{[l:n]})$ (if one exists), then

$$h_{\text{upper}}(D_{[l:n]}) = \min\{\hat{\mu} + (n - l)\Delta, \mu(D_{[l:n]})\}.$$

Proof. We consider the first equality here, and the second follows symmetrically.

Note that $\mu(D_{[1:k]})$ is increasing in k because $x_1 \leq \dots \leq x_n$. By construction of h_{lower} , if for some index k' we have $h_{\text{lower}}(D_{[1:k']}) \leq \mu(D_{[1:k']})$, then $h_{\text{lower}}(D_{[1:k'+1]}) \leq \mu(D_{[1:k'+1]})$. Accordingly,

if we let k_{min} be the first index such that $h_{lower}(D_{[1,k_{min}]}) \leq \mu(D_{[1,k_{min}]})$, then in the case that $k \geq k_{min}$ we must have $h_{lower}(D_{[1:k]}) = \mu(D_{[1:k]})$. If $k < k_{min}$, then we must have $h_{lower}(D_{[1:k]}) > \mu(D_{[1:k]})$, and furthermore $h_{lower}(D_{[1:k']}) > \mu(D_{[1:k']})$ for all $k' \leq k$, which implies that we always decreased by Δ and we get $h_{lower}(D_{[1:k]}) = \hat{\mu} - k\Delta$. \square

We use the explicit forms of h_{lower} and h_{upper} to sandwich the loss in accuracy, by considering the inflection point of $n/3$ and bounding the error from h_{lower} separately for $k \leq n/3$ and for $k \geq n/3$. The analogous result follows symmetrically for h_{upper} .

Lemma 5.11. *Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function with chosen parameters $\hat{\mu}$ and Δ . If $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ is our Sensitivity-Preprocessing Function with parameter Δ , then given any $D = (x_1, \dots, x_n)$,*

$$|g(D) - \mu(D)| \leq \max\{|\hat{\mu} - \mu(D)| - \frac{n}{3}\Delta, 0\} + \sum_{i=1}^n \max\left\{\frac{27|x_i - \mu(D)|}{n} - \Delta, 0\right\}.$$

Proof. If we can instead prove the same upper bounds for both $|h_{lower}(D) - \mu(D)|$ and $|h_{upper}(D) - \mu(D)|$, then the desired bound for $|g(D) - \mu(D)|$ follows from Lemma 5.9. We give the desired bound for $|h_{lower}(D) - \mu(D)|$, and the bound for $|h_{upper}(D) - \mu(D)|$ follows symmetrically.

Again, let k be the largest index such that $h_{lower}(D_{[1:k]}) \geq \mu(D_{[1:k]})$ (if one exists). If $k \leq n/3$ or none exists, then it immediately follows from Lemma 5.10 that $h_{lower}(D) \geq \hat{\mu} + \frac{n}{3}\Delta$, which implies $|\mu(D) - h_{lower}(D)| \leq |\mu(D) - \hat{\mu}| - \frac{n}{3}\Delta$.

If $k \geq n/3$, then it is implied by Lemma 5.10 that $h_{lower}(D) = \max\{\hat{\mu} - k\Delta, \mu(D_{[1:k]})\} + (n - k)\Delta \geq \mu(D_{[1:k]}) + (n - k)\Delta$ and therefore,

$$\mu(D) - h_{lower}(D) \leq \mu(D) - \mu(D_{[1:k]}) + (n - k)\Delta = \sum_{i=k}^{n-1} (\mu(D_{[1:i+1]}) - \mu(D_{[1:i]})) - (n - k)\Delta.$$

Furthermore,

$$\mu(D_{[1:i+1]}) - \mu(D_{[1:i]}) = \frac{x_1 + \dots + x_{i+1}}{i+1} - \frac{x_1 + \dots + x_i}{i} = \frac{1}{i(i+1)} \left(\sum_{j=1}^i x_{i+1} - x_j \right).$$

We use the fact that $i \geq n/3$ to achieve,

$$\mu(D) - h_{lower}(D) \leq \left(\frac{9}{n^2} \sum_{i=k}^n \sum_{j=1}^i (x_i - x_j) \right) - (n - k)\Delta.$$

Applying Lemma 5.12 (stated below) gives,

$$\mu(D) - h_{lower}(D) \leq \left(\frac{27}{n} \sum_{i=k}^n |x_i - \mu(D)| \right) - (n - k)\Delta = \sum_{i=k}^n \left(\frac{27|x_i - \mu(D)|}{n} - \Delta \right)$$

We then add in non-negative terms that are necessary for the symmetric version with h_{upper} to achieve our desired bound. \square

We used the following lemma to simplify the bounds in Lemma 5.11 beyond those stated in the more general Lemma 3.6. We relegate the proof of this lemma to the appendix.

Lemma 5.12. For any set of reals $D = (x_1, \dots, x_n)$ where $x_1 \leq \dots \leq x_n$, given any index $k \in [n]$,

$$\frac{1}{n^2} \sum_{i=k}^n \sum_{j=1}^i \frac{1}{3} |x_i - x_j| \leq \frac{1}{n} \sum_{i=k}^n |x_i - \mu(D)|.$$

To finally obtain all the necessary components for the proof of Theorem 1.7, it is only left to show that when all the inputs of the database are in a nicely bounded range, our Sensitivity-Preprocessing Function will perfectly fit to the function μ .

Lemma 5.13. Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function with chosen parameters $\hat{\mu}$ and Δ . If $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ is our Sensitivity-Preprocessing Function with parameter Δ , then given any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$, if for all $x_i \in D$ we have $x_i \in [\hat{\mu} + \alpha\Delta, \hat{\mu} + (\alpha + n)\Delta]$ for $\alpha \in [-n, 0]$, then $g(D) = \mu(D)$.

Proof. First, it is straightforward to see by the construction of h_{lower} and h_{upper} that $h_{lower}(D) \leq \mu(D)$ if $\mu(D) \geq \hat{\mu} - n\Delta$ and $h_{upper}(D) \geq \mu(D)$ if $\mu(D) \leq \hat{\mu} + n\Delta$. Therefore, by Lemma 5.9, the desired result is implied if $h_{lower}(D) \geq \mu(D)$ and $h_{upper}(D) \leq \mu(D)$. Here we show that $h_{lower}(D) \geq \mu(D)$, and $h_{upper}(D) \leq \mu(D)$ will be implied symmetrically.

Suppose it is not true that $h_{lower}(D) \geq \mu(D)$, then there must exist the last index $k < n$ such that $h_{lower}(D_{[1:k]}) \geq \mu(D_{[1:k]})$, which by construction implies that $h_{lower}(D) = h_{lower}(D_{[1:k]}) + (n - k)\Delta$. To achieve our contradiction, we want to show that $\mu(D) - h_{lower}(D_{[1:k]}) \leq (n - k)\Delta$.

By our restriction of each x_i and by assumption we have,

$$\hat{\mu} + \alpha\Delta \leq \mu(D_{[1:k]}) \leq h_{lower}(D_{[1:k]}).$$

Furthermore, because all of the remaining $x_i \leq \hat{\mu} + (\alpha + n)\Delta$, we must have,

$$\mu(D) \leq \frac{k\mu(D_{[1:k]}) + (n - k)(\hat{\mu} + (\alpha + n)\Delta)}{n} \leq \frac{k \cdot h_{lower}(D_{[1:k]}) + (n - k)(\hat{\mu} + (\alpha + n)\Delta)}{n},$$

where the second inequality follows from our assumption that $h_{lower}(D_{[1:k]}) \geq \mu(D_{[1:k]})$. This implies,

$$\begin{aligned} \mu(D) - h_{lower}(D_{[1:k]}) &\leq \frac{k \cdot h_{lower}(D_{[1:k]}) + (n - k)(\hat{\mu} + (\alpha + n)\Delta)}{n} - h_{lower}(D_{[1:k]}) \\ &= \frac{(k - n)h_{lower}(D_{[1:k]}) + (n - k)(\hat{\mu} + \frac{n}{2}\Delta)}{n} \end{aligned}$$

We use the fact that $h_{lower}(D_{[1:k]}) \geq \hat{\mu} + \alpha\Delta$ and $k < n$ to get,

$$\mu(D) - h_{lower}(D_{[1:k]}) \leq \frac{(k - n)(\hat{\mu} + \alpha\Delta) + (n - k)(\hat{\mu} + (\alpha + n)\Delta)}{n} = (n - k)\Delta,$$

giving our desired contradiction, which implies $h_{lower}(D) \geq \mu(D)$. \square

5.3.1 Proof of Theorem 1.7

We now have all the necessary components to give our proof of Theorem 1.7, which we restate and prove below.

Theorem 1.7. Let $\mu : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the mean function for the data universe of all finite-length real-valued vectors. For chosen parameters $\hat{\mu}$ and Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, we give $O(n^2)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that,

$$|g(D) - \mu(D)| \leq \max \left\{ |\mu(D) - \hat{\mu}| - \frac{n}{3}\Delta, 0 \right\} + \sum_{i=1}^n \max \left\{ \frac{27|x_i - \mu(D)|}{n} - \Delta, 0 \right\}.$$

Additionally, if we are guaranteed that each $x_i \in [\hat{\mu} + \alpha\Delta, \hat{\mu} + (\alpha + n)\Delta]$ for $\alpha \in [-n, 0]$, then $g(D) = \mu(D)$

Proof of Theorem 1.7. The fact that g has sensitivity Δ follows from the fact that it is our Sensitivity-Preprocessing Function and the guarantees of Lemma 3.5. The runtime follows from Corollary 5.3. We then achieve the error bounds from Lemma 5.11 and Lemma 5.13. \square

6 Efficient Implementation for Variance

In this section, we show how to efficiently extend our recursive algorithm to *variance*, which is an important statistical metric and a more complicated function than those considered in Section 5. Although variance is not a database-ordered function, we can still implement our Sensitivity-Preprocessing Function for variance in $O(n^2)$ time, using similar techniques to reduce the number of subproblems that must be considered. This suggests that database-ordered functions are not the only class that have an efficient implementation, and that running time of our algorithm can be improved more generally using structural properties of the function being considered.

The general idea will remain the same as we reduce the number of subproblems to $O(n^2)$ by using structural properties of variance. We first formally define the discrete version of variance with two equivalent equations.

Definition 6.1. For any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$, let $\mu(D) = \frac{1}{n}(x_1 + \dots + x_n)$ and define the variance function,

$$\mathbf{Var}[D] \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n (x_i - \mu(D))^2,$$

or equivalently,

$$\mathbf{Var}[D] \stackrel{\text{def}}{=} \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \frac{1}{2} (x_i - x_j)^2.$$

As with mean, α -trimmed mean, median, maximum, and minimum, we will first sort the entries of the database. Intuitively, we can decrease the variance most by removing either the minimum or maximum value. We make use of the following fact, which we prove in the appendix for completeness.

Fact 6.2. Given $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ such that $x_1 \leq \dots \leq x_n$, then for any i ,

$$\min\{\mathbf{Var}[D - x_1], \mathbf{Var}[D - x_n]\} \leq \mathbf{Var}[D - x_i].$$

We will use this fact to show that the lower bound on $g(D)$ will be defined by $g(D - x_1)$ or $g(D - x_n)$. The difficulty now becomes that to increase variance the most, we would want to remove an entry between x_1 and x_n . This poses a significant complication in constructing

a dynamic program for the subproblems. More specifically, even if $g(D)$ only required solving two subproblems $g(D - x_i)$ and $g(D - x_j)$ for some x_i, x_j , we are still doubling the number of subproblems at each step. The straightforward dynamic program for ordered-databases was able to reuse different subproblems to avoid a runtime blow-up. The key idea will then be that we can bound, with respect to the original variance, the amount variance can be increase by removing an entry. In particular, we use the following fact that is likely a folklore result, but we could not find a citation, so we prove it in the appendix for completeness.

Fact 6.3. *Given any unordered $(x_1, \dots, x_n) \in \mathbb{R}^n$,*

$$\mathbf{Var}[x_1, \dots, x_{n-1}] \leq \frac{n}{n-1} \mathbf{Var}[x_1, \dots, x_n].$$

We can then use this strong bound to show that if we initialize $g(\emptyset) = 0$, the Sensitivity-Preprocessing Function will never go above $\mathbf{Var}[D]$ for any $g(D)$. As a result, the Sensitivity-Preprocessing Function will never actually use $\text{LOWER}(D)$. This will then allow us to only recurse on subproblems where the minimum or maximum has been removed, and the dynamic program will be analogous to the one given for mean.

We first give the efficient implementation for variance and show that it can be done in $O(n^2)$ time. Then we give stronger bounds on the error incurred by this efficient implementation, and finally use these facts to prove Theorem 1.8.

6.1 Efficient algorithm for variance

As with mean and the database-ordered functions, the key to our efficient implementation will be showing that the Sensitivity-Preprocessing Function can be equivalently defined using far fewer subproblems. Using some of the intuition above, we are able to prove the following lemma that reduces the Sensitivity-Preprocessing Function to a much simpler recursion.

Lemma 6.4. *Let $\mathbf{Var} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the variance function and set $\mathbf{Var}[\emptyset] = 0$. Then the Sensitivity-Preprocessing Function with parameter Δ can be equivalently defined as $g(\emptyset) = 0$ and $g(D) = \min\{\mathbf{Var}[D], g(D - x_1) + \Delta, g(D - x_n) + \Delta\}$ where $D = (x_1, \dots, x_n)$ with $x_1 \leq \dots \leq x_n$.*

We will prove this lemma with the following two helper lemmas. The first will show that the Sensitivity-Preprocessing Function will never exceed the true variance. The second uses the fact that variance is minimized by either removing the minimum or maximum value to show that the lower bound can simply consider the subproblems $g(D - x_1)$ and $g(D - x_n)$.

Lemma 6.5. *Given any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$, if g is the Sensitivity-Preprocessing Function of variance with parameter Δ and $g(\emptyset) = 0$, then,*

$$g(D) \leq \mathbf{Var}[D].$$

Proof. We will prove this by induction. If D contains only a single entry, then $\mathbf{Var}[D] = 0$ and by construction $g(D) = 0$.

We then consider $D = (x_1, \dots, x_n)$ and assume the inequality holds for all subsets. By the definition of the Sensitivity-Preprocessing Function, it suffices to show that $g(D - x_i) - \Delta \leq \mathbf{Var}[D]$ for all x_i . Our inductive claim gives $g(D - x_i) \leq \mathbf{Var}[D - x_i]$, and Fact 6.3 implies:

$$\mathbf{Var}[D - x_i] - \mathbf{Var}[D] \leq \frac{1}{n-1} \mathbf{Var}[D],$$

These combine to give,

$$g(D - x_i) - \mathbf{Var}[D] \leq \frac{1}{n-1} \mathbf{Var}[D].$$

We now consider two cases. If $g(D - x_i) \leq \mathbf{Var}[D]$, then $g(D - x_i) - \Delta \leq \mathbf{Var}[D]$ because $\Delta \geq 0$ and we have our desired inequality. If $\mathbf{Var}[D] \leq g(D - x_i)$ then,

$$g(D - x_i) - \mathbf{Var}[D] \leq \frac{1}{n-1} g(D - x_i).$$

Further, by the definition of Sensitivity-Preprocessing Function and the fact that $g(\emptyset) = 0$, we must have $g(D - x_i) \leq (n-1)\Delta$, implying,

$$g(D - x_i) - \mathbf{Var}[D] \leq \Delta,$$

which is our desired inequality. \square

Lemma 6.6. *Given $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ such that $x_1 \leq \dots \leq x_n$ and g is the Sensitivity-Preprocessing Function of variance with parameter Δ and $g(\emptyset) = 0$, then,*

$$\min\{g(D - x_1), g(D - x_n)\} \leq g(D - x_i),$$

for any $x_i \in D$.

Proof. We will prove this by induction. If D has just one entry then $x_1 = x_i = x_n$ and each term is equivalent.

We then consider $D = (x_1, \dots, x_n)$ and assume the inequality holds for all subsets. We will consider two cases. Our first case is $g(D - x_i) = \mathbf{Var}[D - x_i]$. Lemma 6.5 implies:

$$\min\{g(D - x_1), g(D - x_n)\} \leq \min\{\mathbf{Var}[D - x_1], \mathbf{Var}[D - x_n]\}.$$

Furthermore, by Fact 6.2 we have $\min\{\mathbf{Var}[D - x_1], \mathbf{Var}[D - x_n]\} \leq \mathbf{Var}[D - x_i]$. Combining this with the assumption $g(D - x_i) = \mathbf{Var}[D - x_i]$ gives the desired inequality.

It is implied by Lemma 6.5 that the only other case we need to consider is $g(D - x_i) < \mathbf{Var}[D - x_i]$. This assumption and our definition of Sensitivity-Preprocessing Function together imply,

$$g(D - x_i) = \min_{j \neq i} \{g(D - x_i - x_j) + \Delta\}.$$

The definition of Sensitivity-Preprocessing Function also gives:

$$\min\{g(D - x_1), g(D - x_n)\} \leq \min\{\min_{j \neq 1} \{g(D - x_1 - x_j) + \Delta\}, \min_{j \neq n} \{g(D - x_n - x_j) + \Delta\}\}.$$

As a result, if $\min_{j \neq 1} \{g(D - x_1 - x_j) + \Delta\}$ is minimized for $j = 1$ or $j = n$, then we easily have $\min\{g(D - x_1), g(D - x_n)\} \leq g(D - x_i)$. Furthermore, if $j \neq 1, n$, then it suffices to show that,

$$\min\{g(D - x_1 - x_j), g(D - x_n - x_j)\} \leq g(D - x_i - x_j),$$

which follows from the inductive hypothesis and implies our desired result. \square

These two helper lemmas now easily imply Lemma 6.4.

Proof of Lemma 6.4. Lemma 6.5 implies that we will never need to use $\text{LOWER}(D)$, so we can eliminate that case. Further, Lemma 6.6 implies that $\text{UPPER}(D) = \min\{g(D - x_1) + \Delta, g(D - x_n) + \Delta\}$. Combining these facts implies our recursion defined in the lemma statement is equivalent to the Sensitivity-Preprocessing Function. \square

Algorithm 3 Efficient Implementation for Variance

Input: Variance function $\mathbf{Var} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$, sensitivity bound Δ , and database $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ for some arbitrary n .
Output: $g(D)$ where g is the Sensitivity-Preprocessing Function of variance with parameter Δ .
Initialize $g(\emptyset) = 0$
Sort D (We will assume $x_1 \leq \dots \leq x_n$ for simplicity)
for $k=1, \dots, n$ **do**
 for $i = 1, \dots, n-k+1$ **do**
 for every database $D' = (x_i, \dots, x_{i+k-1})$ **do**
 Let $g(D') = \min\{\mathbf{Var}[D'], g(D' - x_i) + \Delta, g(D' - x_{i+k-1}) + \Delta\}$
 end for
 end for
end for
Output $g(D)$

With this reduction in the number of subproblems for the Sensitivity-Preprocessing Function, we will be able to give a similar efficient dynamic programming algorithm for the implementation.

It immediately follows that the number of subproblems that we need to consider is $O(n^2)$, but we still need to efficiently compute $\mathbf{Var}[D]$. This computation would normally take $O(n)$ time and increase our running time to $O(n^3)$. However, we can use the computation from previous subproblems to compute the variance in $O(1)$ time with the following folklore fact that we prove in the appendix.

Fact 6.7. For any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ and any $x_a \neq x_b \in D$,

$$\mathbf{Var}[D] = \left(\frac{n-1}{n}\right)^2 \mathbf{Var}[D - x_a] + \left(\frac{n-1}{n}\right)^2 \mathbf{Var}[D - x_b] - \left(\frac{n-2}{n}\right)^2 \mathbf{Var}[D - x_a - x_b] + \frac{1}{n^2}(x_a - x_b)^2.$$

With this fact we can now show that we implement the Sensitivity-Preprocessing Function for variance with parameter Δ in $O(n^2)$ time.

Lemma 6.8. Let $\mathbf{Var} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the variance function and set $\mathbf{Var}[\emptyset] = 0$. Then Algorithm 3 will compute $g(D)$ for any database of n entries in $O(n^2)$ time where g is the Sensitivity-Preprocessing Function for variance with parameter Δ .

Proof. Correctness of the procedure follows immediately from Lemma 6.4. The running time follows from the fact that we have $O(n^2)$ subproblems and from Fact 6.7 we can compute $\mathbf{Var}[D]$ in $O(1)$ time using the previous subproblems. \square

6.2 Accuracy guarantees for variance implementation

In this section we give stronger bounds on the error incurred by the Sensitivity-Preprocessing Function. The proofs will be similar to those in Section 5.3 for mean, but will be slightly simpler due to that fact that the Sensitivity-Preprocessing Function will never go above the actual variance. As a result, we achieve a simpler form for the error of the Sensitivity-Preprocessing Function with respect to variance in the following lemma.

Lemma 6.9. Given $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ and g that is the Sensitivity-Preprocessing Function of variance with parameter Δ and $g(\emptyset) = 0$, then there must exist some $D' \subseteq D$ such that $g(D) = \mathbf{Var}[D'] + (n - k)\Delta$ for $k = |D'|$.

Proof. We prove this inductively on the size of D and see immediately that the claim holds by construction for D with a single entry.

We then consider $D = (x_1, \dots, x_n)$ and assume that our claim holds for all subsets. From Lemma 6.5 we know that $\mathbf{Var}[D] \geq g(D)$ for all databases. If $\mathbf{Var}[D] = g(D)$, then our claim is immediately implied. If $g(D) < \mathbf{Var}[D]$ then we must have $g(D) = g(D - x_i) + \Delta$ for some x_i . Applying the inductive hypothesis on $g(D - x_i)$ gives our desired claim. \square

With this lemma in hand, the main idea for bounding accuracy is to condition on the size of D' , which we denote k , and give bounds separately for the cases when $k \leq n/2$ and $k \geq n/2$. When k is small we will just bound our error by $\mathbf{Var}[D] - (n - k)\Delta$ and use the fact that $(n - k)\Delta$ is large. When k is large we will look at the loss in accuracy from $\mathbf{Var}[D] - \mathbf{Var}[D']$ where we will bound this by iteratively applying the following lemma.

Lemma 6.10. *For any $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ and any $x_a \in D$, then*

$$\mathbf{Var}[D] - \mathbf{Var}[D - x_a] \leq \frac{1}{n^2} \sum_{i=1}^n (x_a - x_i)^2.$$

Proof. By the definition of variance,

$$\mathbf{Var}[D] - \mathbf{Var}[D - x_a] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \frac{1}{2} (x_i - x_j)^2 - \frac{1}{(n-1)^2} \sum_{i \neq a} \sum_{j \neq a} \frac{1}{2} (x_i - x_j)^2.$$

This reduces to,

$$\mathbf{Var}[D] - \mathbf{Var}[D - x_a] = \frac{1}{n^2} \sum_{i=1}^n (x_a - x_i)^2 - \frac{2n-1}{n^2(n-1)^2} \sum_{i \neq a} \sum_{j \neq a} \frac{1}{2} (x_i - x_j)^2,$$

which gives our desired equality. \square

Recall that we want to use this lemma to bound $\mathbf{Var}[D] - \mathbf{Var}[D']$ where D' is a subset of D with size k . Suppose $D' = (x_1, \dots, x_k)$ and let $D_i = (x_1, \dots, x_i)$ for any i ; we will use the fact that $\mathbf{Var}[D] - \mathbf{Var}[D'] = \sum_{i=k+1}^n \mathbf{Var}[D_i] - \mathbf{Var}[D_{i-1}]$. The above Lemma 6.10 allows us to bound this sum, which will be the key step in our accuracy bounds.

Lemma 6.11. *Given $D = (x_1, \dots, x_n) \in \mathbb{R}^n$ and g that is the Sensitivity-Preprocessing Function of variance with parameter Δ and $g(\emptyset) = 0$, then*

$$|\mathbf{Var}[D] - g(D)| \leq \max \left\{ \mathbf{Var}[D] - \frac{n}{2}\Delta, 0 \right\} + \sum_{i=1}^n \max \left\{ \sum_{j=1}^n \frac{4(x_i - x_j)^2}{n^2} - \Delta, 0 \right\}.$$

Proof. Note that Lemma 6.5 implies $|\mathbf{Var}[D] - g(D)| = \mathbf{Var}[D] - g(D)$. From Lemma 6.9 we know that $g(D) = \mathbf{Var}[D'] + (n - k)\Delta$ for some $D' \subseteq D$ of size k , and we can rewrite $\mathbf{Var}[D] - g(D) = \mathbf{Var}[D] - \mathbf{Var}[D'] - (n - k)\Delta$. If $k \leq n/2$, then

$$\mathbf{Var}[D] - g(D) \leq \mathbf{Var}[D] - \frac{n}{2}\Delta,$$

because $(n - k) \geq n/2$ and $\mathbf{Var}[D'] \geq 0$.

If $k \geq n/2$, then for simplicity we will assume $D' = (x_1, \dots, x_k)$ and address this assumption later. We then let $D_i = (x_1, \dots, x_i)$ for any i and use the fact that $\mathbf{Var}[D] - \mathbf{Var}[D'] = \sum_{i=k+1}^n \mathbf{Var}[D_i] -$

$\mathbf{Var}[D_{i-1}]$. Lemma 6.10 along with the fact that $k \geq n/2$ allows us to then bound this summation as

$$\sum_{i=k+1}^n \mathbf{Var}[D_i] - \mathbf{Var}[D_{i-1}] \leq \frac{4}{n^2} \sum_{i=k+1}^n \sum_{j=1}^n (x_i - x_j)^2$$

We can then use this to achieve (for $D' = (x_1, \dots, x_k)$)

$$\mathbf{Var}[D] - \mathbf{Var}[D'] - (n-k)\Delta \leq \sum_{i=k+1}^n \left(\sum_{j=1}^n \frac{4(x_i - x_j)^2}{n^2} - \Delta \right)$$

At this point we address the assumption that $D' = (x_1, \dots, x_k)$ by simply adding non-negative terms to the summation and ensuring that all of the entries in D' are included in this summation. This gives us,

$$\mathbf{Var}[D] - \mathbf{Var}[D'] - (n-k)\Delta \leq \sum_{i=1}^n \max \left\{ \sum_{j=1}^n \frac{4(x_i - x_j)^2}{n^2} - \Delta, 0 \right\}.$$

Adding both errors for $k \leq n/2$ and $k \geq n/2$ gives our desired bound. \square

6.3 Proof of Theorem 1.8

We now have all the necessary pieces for Theorem 1.8, which we restate and prove here.

Theorem 1.8. *Let $\mathbf{Var} : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ be the variance function for the data universe of all finite-length real-valued vectors. For fixed parameter Δ , along with any database $D = (x_1, \dots, x_n) \in \mathbb{R}^{<\mathbb{N}}$, we have $O(n^2)$ time access to a function $g : \mathbb{R}^{<\mathbb{N}} \rightarrow \mathbb{R}$ with sensitivity Δ such that,*

$$|g(D) - \mathbf{Var}[D]| \leq \max \left\{ \mathbf{Var}[D] - \frac{n}{2}\Delta, 0 \right\} + \sum_{i=1}^n \max \left\{ \sum_{j=1}^n \frac{4(x_i - x_j)^2}{n^2} - \Delta, 0 \right\}.$$

Proof. The fact that g has sensitivity Δ follows from the fact that it is our Sensitivity-Preprocessing Function from Lemma 6.8, and the guarantees of Lemma 3.5. The runtime also follows from Lemma 6.8. We then achieve the error bounds from Lemma 6.11. \square

7 Sensitivity preprocessing for personalized privacy guarantees

In this section, we introduce *personalized differential privacy*, where each individual in a database may receive a different privacy parameter ϵ_i . We show that our Sensitivity-Preprocessing Function is naturally compatible with this privacy notion, and demonstrate the use of sensitivity-bounded functions for achieving personalized privacy guarantees, using the Laplace Mechanism and the Exponential Mechanism as illustrative examples. The notion of personalized privacy has been previously applied to the design of markets for privacy. We demonstrate the use of Sensitivity-Preprocessing Function for this application in Section 7.2, and hope that our results may be useful tools for this well-studied problem in algorithmic economics.

7.1 Personalized differential privacy

We begin by defining *personalized differential privacy*, which extends the standard definition of differential privacy (Definition 2.1) to a setting where different individuals participating in the same computation may experience different, personalized privacy guarantees. Similar definitions have also been used in previous work [JYC15, ESS15, AGK17, LXJJ17]. Recall from Section 2 that two databases are neighboring if they differ in at most one entry. We will say that two databases are *i-neighbors* if they differ only in the *i*-th entry.

Definition 7.1 (Personalized differential privacy). A mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ is $\{\epsilon_i\}$ -*personally differentially private* if for all *i*, for every pair of *i*-neighbors $D, D' \in \mathcal{D}$, and for every subset of possible outputs $\mathcal{S} \subseteq \mathcal{R}$,

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon_i) \Pr[\mathcal{M}(D') \in \mathcal{S}].$$

Note that any $\{\epsilon_i\}$ -personally differentially private algorithm is also $(\max_i \epsilon_i)$ -differentially private, since differential privacy provides a worst-case guarantee over all pairs of neighboring databases.

In this section, we show that personalized differential privacy can be achieved by combining our sensitivity preprocessing step with existing differentially private mechanisms. An analyst can first apply our preprocessing step to get g with desired individual sensitivity bounds, and then evaluate g using a differentially private algorithm. The resulting $\{\epsilon_i\}$ -personal differential privacy guarantees will depend on the chosen sensitivity parameters $\{\Delta_i\}$. Since the function g is independent of the database, the sensitivity preprocessing step does not leak any additional privacy.

Individual sensitivity guarantees are critical for accurate analysis in this new privacy model. Using only global sensitivity bounds Δ , personally differentially private mechanisms add noise that scales with $\max_i \{\Delta/\epsilon_i\}$. This alone cannot offer significant accuracy improvements because the noise must still scale inversely proportionally to the smallest ϵ_i . By utilizing individual sensitivity bounds, an analyst can tune each Δ_i to scale with ϵ_i to achieve overall accuracy improvements with personalized differential privacy.

We note that *local differential privacy* [KLN⁺08] also affords different privacy guarantees to different individuals in the same database, by perturbing each user’s data locally before submitting it to the database. Significantly stronger accuracy guarantees are possible in the presence of a trusted curator—which we assume in our model—because the analyst can leverage correlation of noise across individuals [Ull18].

A formal statement of the privacy and accuracy guarantees that arise from applying differentially private algorithms to sensitivity-bounded functions will depend on the exact algorithm used. We illustrate this approach below applying it on two of the most foundational differentially private algorithms: the Laplace Mechanism and the Exponential Mechanism.

Laplace Mechanism

The *Laplace Mechanism* [DMNS06] is perhaps the most fundamental of all differentially private algorithms. It first evaluates a real-valued function f on an input database D , and then perturbs the answer by adding Laplace noise scaled to the global sensitivity of f divided by ϵ . The *Laplace distribution* with scale b , denoted $\text{Lap}(b)$, has probability density function:

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

Definition 7.2 (Laplace Mechanism [DMNS06]). Given any function $f : \mathcal{D} \rightarrow \mathbb{R}$, the *Laplace Mechanism* is defined as,

$$\mathcal{M}_L(D, f, \Delta f/\epsilon) = f(D) + Y,$$

where Y is drawn from $\text{Lap}(\Delta f/\epsilon)$.⁶

The Laplace Mechanism is ϵ -differentially private [DMNS06]. We now show how to combine the Laplace Mechanism with our Sensitivity-Preprocessing Function to achieve personalized differential privacy guarantees.

Proposition 7.3. *Let $g : \mathcal{D} \rightarrow \mathbb{R}$ be a function with individual sensitivities $\{\Delta_i\}$. For any $\{\epsilon_i\}$, the Laplace Mechanism $\mathcal{M}_L(D, g, \max_j \{\Delta_j/\epsilon_j\})$ is $\{\epsilon_i\}$ -personally differentially private.*

Proof. Let $D, D' \in \mathcal{D}$ be i -neighbors, let $g : \mathcal{D} \rightarrow \mathbb{R}$ be a function with individual sensitivities $\{\Delta_i\}$, and let $r \in \mathbb{R}$ be arbitrary.

$$\begin{aligned} \frac{\Pr[\mathcal{M}_L(D, g, \max_j \{\Delta_j/\epsilon_j\}) = r]}{\Pr[\mathcal{M}_L(D', g, \max_j \{\Delta_j/\epsilon_j\}) = r]} &= \frac{\exp\left(-\min_j \left\{\frac{\epsilon_j}{\Delta_j}\right\} |g(D) - r|\right)}{\exp\left(-\min_j \left\{\frac{\epsilon_j}{\Delta_j}\right\} |g(D') - r|\right)} \\ &= \exp\left(\min_j \left\{\frac{\epsilon_j}{\Delta_j}\right\} (|g(D') - r| - |g(D) - r|)\right) \\ &\leq \exp\left(\min_j \left\{\frac{\epsilon_j}{\Delta_j}\right\} (|g(D) - g(D')|)\right) \\ &\leq \exp\left(\min_j \left\{\frac{\epsilon_j}{\Delta_j}\right\} \Delta_i\right) \\ &\leq \exp(\epsilon_i) \end{aligned}$$

Then this version of the Laplace Mechanism run on a function with individual sensitivities $\{\Delta_i\}$ is $\{\epsilon_i\}$ -personally differentially private. \square

Proposition 7.3 shows that to achieve personalized privacy guarantees for a given function f , one can apply our Sensitivity-Preprocessing Function to produce Sensitivity-Bounded g , and then apply the Laplace Mechanism. The accuracy guarantees of this procedure will depend on the worst-case ratio of Δ_i/ϵ_i , as well as global sensitivity of the original function f . If one person j requires significantly higher privacy protections than the rest of the population, the analyst can account for this by reducing Δ_j . This may greatly improve accuracy over the standard approach, which would require the analyst to add increased noise to the entire population. We address this challenge more concretely in Section 7.2, using the application of market design for private data.

Exponential Mechanism

The *Exponential Mechanism* [MT07] is a powerful private mechanism for answering non-numeric queries with an arbitrary range, such as selecting the best outcome from a set of alternatives. The quality of an outcome is measured by a *score function* $q : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$, which relates each alternative to the underlying data through a real-valued score. The global sensitivity of the score function is

⁶We note that the standard definition of the Laplace Mechanism in [DMNS06] takes ϵ as input instead of $\frac{\Delta f}{\epsilon}$. We use the latter here for ease of notation when extending to personalized differential privacy. This change does not affect the algorithm at all.

measured only with respect to the database argument; it can be arbitrarily sensitive in its range argument:

$$\Delta q = \max_{r \in \mathcal{R}} \max_{D, D' \text{ } i\text{-neighbors}} |q(D, r) - q(D', r)|.$$

We define the individual sensitivity of a quality score analogously with respect to only its database argument:

$$\Delta_i(q) = \max_{r \in \mathcal{R}} \max_{D, D' \text{ } i\text{-neighbors}} |q(D, r) - q(D', r)|.$$

The Exponential Mechanism samples an output from the range \mathcal{R} with probability exponentially weighted by score. Outcomes with higher scores are exponentially more likely to be selected, thus ensuring both privacy and a high quality outcome.

Definition 7.4 (Exponential Mechanism [MT07]). Given a quality score $q : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$, the *Exponential Mechanism* is defined as:⁷

$$\mathcal{M}_E(D, q, \Delta q / \epsilon) = \text{output } r \in \mathcal{R} \text{ with probability proportional to } \exp\left(\frac{\epsilon q(D, r)}{2\Delta q}\right).$$

The Exponential Mechanism is ϵ -differentially private [MT07]. We now show that when a score function has bounded individual sensitivity, the Exponential Mechanism is personally differentially private.

Proposition 7.5. *Let $q : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$ be a score function with individual sensitivities $\{\Delta_i\}$. For any $\{\epsilon_i\}$, the Exponential Mechanism $\mathcal{M}_E(D, q, \max_j \{\Delta_j / \epsilon_j\})$ is $\{\epsilon_i\}$ -personally differentially private.*

Proof. Let $D, D' \in \mathcal{D}$ be i -neighbors, let q be a score function with individual sensitivities $\{\Delta_i\}$, and let $r \in \mathcal{R}$ be an arbitrary element of the output range.

$$\begin{aligned} & \frac{\Pr[\mathcal{M}_E(D, q, \max_j \{\Delta_j / \epsilon_j\}) = r]}{\Pr[\mathcal{M}_E(D', q, \max_j \{\Delta_j / \epsilon_j\}) = r]} \\ &= \frac{\left(\frac{\exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r) / 2\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r') / 2\right)} \right)}{\left(\frac{\exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D', r) / 2\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D', r') / 2\right)} \right)} \\ &= \left(\frac{\exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r) / 2\right)}{\exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D', r) / 2\right)} \right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D', r') / 2\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r') / 2\right)} \right) \\ &= \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} (q(D, r) - q(D', r)) / 2\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D', r') / 2\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r') / 2\right)} \right) \\ &\leq \exp\left(\frac{1}{2} \min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} \Delta_i\right) \cdot \exp\left(\frac{1}{2} \min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} \Delta_i\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r') / 2\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} q(D, r') / 2\right)} \right) \\ &= \exp\left(\min_j \left\{ \frac{\epsilon_j}{\Delta_j} \right\} \Delta_i\right) \\ &\leq \exp(\epsilon_i) \end{aligned}$$

⁷As with the Laplace Mechanism, we define the Exponential Mechanism to take $\frac{\Delta q}{\epsilon}$ as input, instead of ϵ . This change is purely notational, and has no impact on the algorithm.

□

Remark 7.6. The Exponential Mechanism is a canonical ϵ -differentially private algorithm: every ϵ -differentially private algorithm \mathcal{M} can be written as an instantiation of the Exponential Mechanism using quality score $q(D, r) = \ln(\Pr[\mathcal{M}(D) = r])$ with global sensitivity $\Delta q = \epsilon$. We can use this reduction to show that *any* ϵ -differentially private algorithm can be modified to give personal privacy guarantees using our Sensitivity-Preprocessing Function. First, re-write private mechanism \mathcal{M} as an Exponential Mechanism \mathcal{M}_E , and then perform Sensitivity-Preprocessing on the quality score of \mathcal{M}_E . Proposition 7.5 shows that the sensitivity-bounded version of \mathcal{M}_E will satisfy personalized differential privacy.

7.2 Application: Markets for privacy

One motivating application for wanting personalized privacy guarantees comes from algorithmic game theory and the study of market design for privacy. This is a well-studied problem in the algorithmic economics community [CCK⁺13, NOS12, NST12, LR12, FL12, GR15, GLRS14, CLR⁺15, CIL15, WFA15, CPWV16], and of practical importance as growing amounts of data are collected about individuals. In a *market for privacy*, a data analyst wishes to purchase and aggregate data from multiple strategic individuals. These individuals may have privacy concerns, and will require compensation for their privacy loss from sharing data. On the opposite side of the market, firms demand accurate estimates of population statistics, for uses such as market research or operational decision making.

The analyst must first purchase data from these strategic individuals, and then aggregate the collected data into an accurate estimate for firms. Her goal is to perform this task while maximizing her own profits. One of the tools at her disposal is differential privacy: by offering individuals formal privacy guarantees, their privacy costs from sharing data are diminished, and the analyst can provide smaller payments. However, the noise from differential privacy may introduce additional error.

It is the analyst’s task to determine the optimal privacy level for the market that balances these opposing effects. Due to potentially heterogeneous privacy costs of the individuals, it may be optimal in terms of her profit for the analyst to provide different privacy guarantees to different individuals in the population. She could then use our Sensitivity-Preprocessing Function to algorithmically provide the heterogeneous privacy levels demanded by the market. We leave the challenge of modeling specifics of these markets as an open question to the algorithmic game theory community, and hope that our preprocessing tool and mechanisms for personalized privacy will open new avenues for designing markets for privacy.

8 Extension to 2-dimensions for ℓ_1 sensitivity

In this section we show that our Sensitivity-Preprocessing Function can be naturally extended to functions that map to 2-dimensional space where we consider the sensitivity in the ℓ_1 distance metric. While there is a natural extension of our Sensitivity-Preprocessing Function to higher dimensions, the primary difficulty will be ensuring that our greedy construction still yields a non-empty intersection of the constraints. Interestingly, we show that this set of constraints will give a non-empty intersection for 2 dimensions, and provide a counter-example for higher dimensions.

Recall that our Sensitivity-Preprocessing Function found a range $[\text{LOWER}(D), \text{UPPER}(D)]$ where it could feasibly place $g(D)$, then choose the point in that segment closest to $f(D)$. This range of feasible points came from intersecting each constraint $[g(D - x_i) - \Delta_i, g(D - x_i) + \Delta_i]$ induced by

the neighbors of D that are strictly smaller. The Sensitivity-Preprocessing Function then chose the point in this intersection closest to $f(D)$. The key property needed by the algorithm was that this intersection was non-empty.

To prove this key property we took advantage of the data universe structure, which immediately yielded the fact that for any $x_i, x_j \in D$ we must have $[g(D - x_i) - \Delta_i, g(D - x_i) + \Delta_i] \cap [g(D - x_j) - \Delta_j, g(D - x_j) + \Delta_j] \neq \emptyset$. As a result, we had a finite set of line segments whose intersection was pair-wise non-empty, which immediately implies that the intersection of all line segments was non-empty. We note that $[g(D - x_i) - \Delta_i, g(D - x_i) + \Delta_i]$ is the ℓ_1 ball with radius Δ_i around $g(D - x_i)$ in one dimension. For higher dimensions, the constraints will now be the ℓ_1 ball with radius Δ_i around $g(D - x_i)$ in d dimensions. The structure of the data universe will still give that each of these ℓ_1 balls has a non-empty pair-wise intersection. However, this only implies that the intersection of all these ℓ_1 balls is non-empty if we are in 2 dimensions. We first formally define the notion of an ℓ_1 ball in higher dimensions.

Definition 8.1 (ℓ_1 ball). The ℓ_1 ball around point $x^* \in \mathbb{R}^d$ with radius Δ is the set:

$$(x^*, \Delta)_1^d \stackrel{\text{def}}{=} \{x \in \mathbb{R}^d \mid \|x - x^*\|_1 \leq \Delta\}.$$

To ensure that our choice of $g(D)$ does not violate the individual sensitivity parameter Δ_i , we must place $g(D) \in (g(D - x_i), \Delta_i)_1^d$. In the one-dimensional case, we had the same constraints, but they were simpler to handle because the ℓ_1 ball is simply a line segment. We now define our Sensitivity-Preprocessing Function for two dimensions, which chooses the point that satisfies our constraints and is closest to $f(D)$, just as in the one-dimensional case.

Definition 8.2 (2-dimensional Sensitivity-Preprocessing Function). Given a function $f : \mathcal{D} \rightarrow \mathbb{R}^2$ for any data universe such that for any $D \in \mathcal{D}$, all $D' \subset D$ are also in \mathcal{D} . For any non-negative individual sensitivity parameters $\{\Delta_i\}$, we say that a function $g : \mathcal{D} \rightarrow \mathbb{R}^2$ is a Sensitivity-Preprocessing Function of f with parameters $\{\Delta_i\}$ if $g(\emptyset) = f(\emptyset)$ and

$$g(D) = \text{closest point in } \cap_{x_i \in D} (g(D - x_i), \Delta_i)_1^2 \text{ to } f(D) \text{ in the } \ell_2 \text{ metric.}$$

If all $\Delta_i = \Delta$ for some non-negative Δ , then we say that g is a Sensitivity-Preprocessing Function of f with parameter Δ .

Our primary goal of this section will then be to prove the following theorem that is equivalent to Theorem 3.3 but works for 2-dimensions. We will also point out the key spot within the proof where it breaks for dimensions greater than 2.

Theorem 8.3. *Given $T(n)$ -time query access to an arbitrary $f : \mathcal{D} \rightarrow \mathbb{R}^2$, and sensitivity parameters $\{\Delta_i\}$, we provide $O((T(n) + n)2^n)$ time access to Sensitivity-Preprocessing Function $g : \mathcal{D} \rightarrow \mathbb{R}^2$ such that $\Delta_i(g) \leq \Delta_i$. Further, for any database $D = (x_1, \dots, x_n)$,*

$$\|f(D) - g(D)\|_1 \leq \max_{\sigma \in \sigma_D} \sum_{i=1}^{|D|} \max\{\|f(D_{\sigma(<i)} + x_{\sigma(i)}) - f(D_{\sigma(<i)})\|_1 - \Delta_{\sigma(i)}, 0\},$$

where σ_D is the set of all permutations on $[n]$, and $D_{\sigma(<i)} = (x_{\sigma(1)}, \dots, x_{\sigma(i-1)})$ is the subset of D that includes all individual data in the permutation before the i th entry.

As before, we will break the proof of this theorem into two parts. It immediately follows from construction that our 2-dimensional Sensitivity-Preprocessing Function will have the appropriate individual sensitivity parameters, but only if the function is well-defined. To this end, we first show in Section 8.1 that the intersection of the ℓ_1 balls is always non-empty if each pair-wise intersection is non-empty. Then in Section 8.2 we give the analogous error guarantees where the proof will just follow equivalently to the one-dimensional case.

8.1 Correctness of Sensitivity-Preprocessing Function

In this section we show that for our 2-dimensional Sensitivity-Preprocessing Function, it is always the case that $g(D)$ is defined. This is equivalent to showing:

$$\bigcap_{x_i \in D} (g(D - x_i), \Delta_i)_1^2 \neq \emptyset.$$

We will first take advantage of the structure of data universes to show that the pair-wise intersection is always non-empty. Then we will use the fact that pair-wise intersection of ℓ_1 balls in 2-dimensions implies that the intersection of all ℓ_1 balls is non-empty. Intuitively, this is because ℓ_1 balls in 2-dimensions are simply rotated squares. Further, we will show that this is exactly the step that breaks the algorithm for higher dimensions.

Lemma 8.4. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}^2$ and desired sensitivity parameters $\{\Delta_i\}$, let $g : \mathcal{D} \rightarrow \mathbb{R}^2$ be the Sensitivity-Preprocessing Function with parameters $\{\Delta_i\}$. For any $D \in \mathcal{D}$ with at least two entries, assume that $g(D')$ is defined for any $D' \subset D$. Then for any $x_i, x_j \in D$,*

$$(g(D - x_i), \Delta_i)_1^2 \cap (g(D - x_j), \Delta_j)_1^2 \neq \emptyset.$$

Note that we have not yet proven that $g(D)$ is defined on all databases, so we will need to first assume that it is on all subsets of D . Our proof of this fact will be done inductively.

Proof. We use the fact that D has at least two entries and consider the database $D - x_i - x_j$. Due to our assumption that $g(D')$ is defined on all $D' \subset D$, it follows from our construction of g that

$$\|g(D - x_i) - g(D - x_i - x_j)\|_1 \leq \Delta_j,$$

and

$$\|g(D - x_j) - g(D - x_i - x_j)\|_1 \leq \Delta_i,$$

Applying triangle inequality gives,

$$\|g(D - x_i) - g(D - x_j)\|_1 \leq \Delta_i + \Delta_j,$$

which implies our claim by the definition of ℓ_1 balls. \square

With this pair-wise intersection property, it now remains to be shown that this implies the intersection of all ℓ_1 balls is non-empty. For this we prove a general fact about the intersection of ℓ_1 balls in 2-dimensions.

Lemma 8.5. *Consider any set of points $y_1, \dots, y_n \in \mathbb{R}^2$, where we let $(y_i)_1$ and $(y_i)_2$ denote the respective coordinates of y_i . Consider any set of non-negative $\Delta_1, \dots, \Delta_n$. If for any y_i, y_j ,*

$$(y_i, \Delta_i)_1^2 \cap (y_j, \Delta_j)_1^2 \neq \emptyset,$$

then,

$$\bigcap_{i=1}^n (y_i, \Delta_i)_1^2 \neq \emptyset.$$

Our proof will first rewrite each ℓ_1 ball as a set of 4 linear inequalities. From this interpretation we will then use two critical facts. First, each inequality has a corresponding parallel inequality in any other ℓ_1 ball. Second, removing any one of these constraints gives an unbounded polytope.

Proof. Further examination of Definition 8.1 shows that

$$(y_i, \Delta_i)_1^2 \stackrel{\text{def}}{=} \{x \in \mathbb{R}^2 \mid \|x - y_i\|_1 \leq \Delta_i\} = \{x \in \mathbb{R}^2 \mid |(x)_1 - (y_i)_1| + |(x)_2 - (y_i)_2| \leq \Delta_i\}.$$

We then use a known trick of converting absolute values into linear inequalities where $|x| \leq k$ becomes $x \leq k$ and $-x \leq k$.

$$\begin{aligned} (y_i, \Delta_i)_1^2 &= \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq (y_i)_1 + (y_i)_2 + \Delta_i\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid -(x)_1 - (x)_2 \leq -(y_i)_1 - (y_i)_2 + \Delta_i\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid (x)_1 - (x)_2 \leq (y_i)_1 - (y_i)_2 + \Delta_i\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid -(x)_1 + (x)_2 \leq -(y_i)_1 + (y_i)_2 + \Delta_i\} \end{aligned}$$

At this point we note that each of the balls have parallel inequalities, so we can use the following fact:

$$\begin{aligned} \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq (y_i)_1 + (y_i)_2 + \Delta_i\} \cap \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq (y_j)_1 + (y_j)_2 + \Delta_j\} \\ = \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq \min\{(y_i)_1 + (y_i)_2 + \Delta_i, (y_j)_1 + (y_j)_2 + \Delta_j\}\}. \end{aligned}$$

We apply this fact to the full intersection and obtain,

$$\begin{aligned} \bigcap_{i=1}^n (y_i, \Delta_i)_1^2 &= \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq \min_{i \in [n]} \{(y_i)_1 + (y_i)_2 + \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid -(x)_1 - (x)_2 \leq \min_{i \in [n]} \{-(y_i)_1 - (y_i)_2 + \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid (x)_1 - (x)_2 \leq \min_{i \in [n]} \{(y_i)_1 - (y_i)_2 + \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid -(x)_1 + (x)_2 \leq \min_{i \in [n]} \{-(y_i)_1 + (y_i)_2 + \Delta_i\}\}. \end{aligned}$$

Intuitively, if this intersection exists, it must be a rectangle that is rotated 45 degrees. To more easily see this fact, we now multiply the second and fourth constraint by -1.

$$\begin{aligned} \bigcap_{i=1}^n (y_i, \Delta_i)_1^2 &= \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \leq \min_{i \in [n]} \{(y_i)_1 + (y_i)_2 + \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid (x)_1 + (x)_2 \geq \min_{i \in [n]} \{(y_i)_1 + (y_i)_2 - \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid (x)_1 - (x)_2 \leq \min_{i \in [n]} \{(y_i)_1 - (y_i)_2 + \Delta_i\}\} \\ &\quad \cap \{x \in \mathbb{R}^2 \mid (x)_1 - (x)_2 \geq \min_{i \in [n]} \{(y_i)_1 - (y_i)_2 - \Delta_i\}\} \end{aligned}$$

With this interpretation it is straightforward to see that $\bigcap_{i=1}^n (y_i, \Delta_i)_1^2 = \emptyset$ if and only if

$$\min_{i \in [n]} \{(y_i)_1 + (y_i)_2 + \Delta_i\} < \min_{i \in [n]} \{(y_i)_1 + (y_i)_2 - \Delta_i\},$$

or

$$\min_{i \in [n]} \{(y_i)_1 - (y_i)_2 + \Delta_i\} < \min_{i \in [n]} \{(y_i)_1 - (y_i)_2 - \Delta_i\}.$$

Let k be the index that minimizes $(y_i)_1 + (y_i)_2 + \Delta_i$ and let l be the index that minimizes $(y_i)_1 + (y_i)_2 - \Delta_i$. If

$$(y_k)_1 + (y_k)_2 + \Delta_k < (y_l)_1 + (y_l)_2 - \Delta_l,$$

then we must have,

$$\Delta_k + \Delta_l < (y_l)_1 - (y_k)_1 + (y_l)_2 - (y_k)_2 \leq |(y_l)_1 - (y_k)_1| + |(y_l)_2 - (y_k)_2|,$$

which contradicts our assumption that $(y_k, \Delta_k)_1^2 \cap (y_l, \Delta_l)_1^2 \neq \emptyset$. This follows identically for the second inequality, so therefore neither of them can hold and the intersection must be non-empty. \square

We now remark that Theorem 8.3 cannot be extended to higher dimensions or to ℓ_p norms.

Remark 8.6. For extending to dimensions greater than two, the proof breaks down at Lemma 8.5. Intuitively, we can still interpret each ℓ_1 ball as a set of linear inequalities, however it no longer has the critical property that removing one of the constraints creates an unbounded polytope. More specifically, consider the following counter-example for 3 dimensions:

Let $A = \{(1, 1, -1), (1, -1, 1), (-1, 1, 1), (1, -1, -1), (-1, 1, -1), (-1, -1, 1)\}$ and set $\Delta = 3$. It is not difficult to see that taking the intersection of Δ -radius ℓ_1 balls around each point in A will only contain the origin $(0, 0, 0)$. We then consider adding the point $(3/2, 3/2, 3/2)$, and it is straightforward to verify that the ℓ_1 distance between this point and any point in A is at most $11/2 < 2\Delta$. However, the origin is not within the ℓ_1 ball around $(3/2, 3/2, 3/2)$. Therefore, if we consider the set of ℓ_1 balls of radius Δ around the points in $A \cup (3/2, 3/2, 3/2)$, then each pair of ℓ_1 balls will intersect, but the full intersection will be empty, giving our counter-example.

Remark 8.7. Even in 2-dimensions, we cannot have Lemma 8.5 for the ℓ_p ball with $p \in (1, \infty)$ due to the curvature of each ball. For instance, consider the ℓ_2 ball with radius 1 for the points $(-1, 0), (1, 0), (0, \sqrt{3})$. Each of pair of these points is exactly distance 2 apart in the ℓ_2 metric, so their ℓ_2 balls of radius 1 each pairwise intersect. However it is easy to see that the intersection of all three is empty.

We can similarly extend this counter-example to other ℓ_p balls using the fact that there must be some curvature of the ℓ_p ball, and the midpoint between any two points in the ℓ_p metric is unique if $p \in (1, \infty)$.

With these lemmas, we are now able to show that our 2-dimensional Sensitivity-Preprocessing Function must always be defined.

Lemma 8.8. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}^2$ with sensitivity parameters $\{\Delta_i\}$, let $g : \mathcal{D} \rightarrow \mathbb{R}^2$ be the Sensitivity-Preprocessing Function with parameters $\{\Delta_i\}$. Then for any $D \in \mathcal{D}$,*

$$\bigcap_{x_i \in D} (g(D - x_i), \Delta_i)_1^2 \neq \emptyset.$$

Proof. We will prove this fact inductively, and note that it is immediately true when D only has one entry.

We then consider an arbitrary database D and assume that it is true for all $D' \subset D$. With this inductive claim we can apply Lemma 8.4 to get that all of the ℓ_1 balls have non-empty pairwise intersection. Our desired result then immediately follows from applying Lemma 8.5. \square

8.2 Error bounds for the 2-dimensional extension

The following lemma gives the desired error bounds on the 2-dimensional Sensitivity-Preprocessing Function.

Lemma 8.9. *Given any $f : \mathcal{D} \rightarrow \mathbb{R}^2$ and desired sensitivity parameters $\{\Delta_i\}$, let $g : \mathcal{D} \rightarrow \mathbb{R}^2$ be the Sensitivity-Preprocessing Function with parameters $\{\Delta_i\}$. Then for any $D \in \mathcal{D}$,*

$$\|f(D) - g(D)\|_1 \leq \max_{\sigma \in \sigma_D} \sum_{i=1}^{|D|} \max\{\|f(D_{\sigma(<i)}) + x_{\sigma(i)} - f(D_{\sigma(<i)})\|_1 - \Delta_{\sigma(i)}, 0\},$$

where σ_D is the set of all permutations of the set $[n]$, and let $D_{\sigma(<i)} = (x_{\sigma(1)}, \dots, x_{\sigma(i-1)})$ be the subset of D that includes all individual data in the permutation before the i th entry.

The proof of Lemma 8.9 follows identically to the proof of Lemma 3.6 where by replacing any instance of absolute value with the 1-norm.

We are finally ready to complete the proof of our main theorem for two dimensions.

Proof of Theorem 8.3. The individual sensitivity guarantees follow from the construction of g and Lemma 8.8. The error bounds are given by Lemma 8.9. It then remains to prove the running time. For each subset of D we need to query f which takes $T(n)$ time by assumption. We note that within the proof of Lemma 8.5 we gave a construction for obtaining the intersection of n different ℓ_1 balls which could clearly be done in $O(n)$ time. Finding the closest point to $f(D)$ then takes $O(1)$ time for the polytope defined by four inequalities. Therefore, the running time is $T(n) + O(n)$ for each of the 2^n subsets, which implies the desired running time. \square

9 Future Directions

We are especially interested in efficiently implementing our framework for more complicated and, in particular, higher-dimensional functions such as linear regression. We believe that leveraging the simple recursive construction of our algorithm along with non-trivial structural properties of these more difficult functions can allow for efficient and accurate implementation. We are particularly optimistic because all of our proofs in this work were from first principles, suggesting that we may be able to obtain further results from this framework by using more sophisticated tools.

While our construction did not generalize to any dimension under the ℓ_1 sensitivity metric, we note that this was in the most general setting. If the class of functions we consider is significantly restricted, then we believe the natural extension could both work and be efficiently implementable. Furthermore, we have not yet investigated variants of our algorithm that might work better under stronger assumptions or combining our construction with other frameworks for handling worst-case sensitivity.

We also believe that our construction opens up several intriguing directions with respect to personalized differential privacy and its application in markets for privacy. Our construction allows for tailoring individual sensitivity, but this presents a natural trade-off between choosing small individual sensitivity parameters and the error incurred by our preprocessing step. For specific functions, this may yield interesting optimization problems that can also be considered in the context of markets for privacy.

Acknowledgements

We thank Richard Peng, Aaron Roth, and Jamie Morgenstern for useful feedback and discussions. We also thank the participants of the Banff International Research Station Workshop on Mathematical Foundations of Data Privacy—especially Adam Smith—for comments and discussion on an earlier draft of this paper.

References

- [AGK17] Mohammad Alaggan, Sebastien Gambs, and Anne-Marie Kermarrec. Heterogeneous differential privacy. *Journal of Privacy and Confidentiality*, 7(6):127–158, 2017.
- [AKZ⁺17] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. BLENDER: Enabling local search with a hybrid differential privacy model. In *26th USENIX Security Symposium*, USENIX Security '17, pages 747–764, 2017.
- [BBDS13] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 87–96, 2013.
- [BF16] Raef Bassily and Yoav Freund. Typicality-based stability and privacy. *CoRR*, abs/1604.03336, 2016.
- [BLR08] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC*, pages 609–618, 2008.
- [CCK⁺13] Yiling Chen, Stephen Chong, Ian A. Kash, Tal Moran, and Salil Vadhan. Truthful mechanisms for agents that value privacy. In *Proceedings of the 14th ACM Conference on Electronic Commerce*, EC '13, pages 215–232, 2013.
- [CIL15] Rachel Cummings, Stratis Ioannidis, and Katrina Ligett. Truthful linear regression. In *Proceedings of The 28th Conference on Learning Theory*, COLT '15, pages 448–483, 2015.
- [CLN⁺16] Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven Wu. Adaptive learning with robust generalization guarantees. In *29th Annual Conference on Learning Theory*, COLT '16, pages 772–814, 2016.
- [CLR⁺15] Rachel Cummings, Katrina Ligett, Aaron Roth, Zhiwei Steven Wu, and Juba Ziani. Accuracy for sale: Aggregating data with a variance constraint. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 317–324, 2015.
- [CPWV16] Rachel Cummings, David M. Pennock, and Jennifer Wortman Vaughan. The possibilities and limitations of private prediction markets. In *Proceedings of the 17th ACM Conference on Economics and Computation*, EC '16, pages 143–160, 2016.
- [CZ13] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2013.
- [DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st ACM Symposium on Theory of Computing*, STOC '09, 2009.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, 2006.

- [DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(34):211–407, 2014.
- [ESS15] Hamid Ebadi, David Sands, and Gerardo Schneider. Differential privacy: Now it’s getting personal. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL ’15, pages 69–81, 2015.
- [FL12] Lisa Fleischer and Yu-Han Lyu. Approximately optimal auctions for selling privacy when costs are correlated with data. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC ’12, pages 568–585, 2012.
- [GLRS14] Arpita Ghosh, Katrina Ligett, Aaron Roth, and Grant Schoenebeck. Buying private data without verification. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC ’14, pages 931–948, 2014.
- [GR15] Arpita Ghosh and Aaron Roth. Selling privacy at auction. *Games and Economic Behavior*, 91:334–346, 2015. Preliminary Version appeared in the Proceedings of the 12th ACM Conference on Electronic Commerce (EC 2011).
- [HSR⁺08] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLOS Genetics*, 4(8):1–9, 08 2008.
- [JYC15] Zach Jorgensen, Ting Yu, and Graham Cormode. Conservative or liberal? Personalized differential privacy. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 1023–1034, 2015.
- [KLN⁺08] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS ’08, pages 531–540, 2008.
- [KLSU] Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan Ullman. Privately learning high-dimensional distributions. arXiv preprint 1805.00216.
- [KNRS13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, TCC ’13, pages 457–476, 2013.
- [LR12] Katrina Ligett and Aaron Roth. Take it or leave it: Running a survey when privacy comes at a cost. In *Proceedings of the 8th International Conference on Internet and Network Economics*, WINE ’12, pages 378–391, 2012.
- [LXJJ17] Haoran Li, Li Xiong, Zhanglong Ji, and Xiaoqian Jiang. Partitioning-based mechanisms under personalized differential privacy. In *Advances in Knowledge Discovery and Data Mining*, PAKDD ’17, pages 615–627, 2017.
- [MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2007.

- [NOS12] Kobbi Nissim, Claudio Orlandi, and Rann Smorodinsky. Privacy-aware mechanism design. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC '12*, pages 774–789. ACM, 2012.
- [NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC '07*, pages 75–84, 2007.
- [NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy, SP '08*, pages 111–125, 2008.
- [NST12] Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. Approximately optimal mechanism design via differential privacy. In *Proceedings of the 2012 Conference on Innovations in Theoretical Computer Science, ITCS '12*, pages 203–213, 2012.
- [RS] Sofya Raskhodnikova and Adam Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. arXiv preprint 1504.07912.
- [RS16] Sofya Raskhodnikova and Adam D. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science, FOCS '16*, pages 495–504, 2016.
- [Ull18] Jonathan Ullman. Tight lower bounds for locally differentially private selection. arXiv preprint 1802.02638, 2018.
- [WFA15] Bo Waggoner, Rafael Frongillo, and Jacob Abernethy. A market framework for eliciting private data. In *Advances in Neural Information Processing Systems 29, NIPS '15*, pages 3492–3500, 2015.

A Omitted Proofs

In this appendix we provide proofs that were omitted from Sections 5 and 6.

A.1 Proof of Lemma 5.12

Proof of Lemma 5.12. We start by decomposing the RHS:

$$\frac{1}{n} \sum_{i=k}^n |x_i - \mu_D| = \frac{1}{n} \sum_{i=k}^n \left| x_i - \frac{x_1 + \dots + x_n}{n} \right| = \frac{1}{n^2} \sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right|.$$

It now suffices to show,

$$\sum_{i=k}^n \sum_{j=1}^i \frac{1}{3} |x_i - x_j| \leq \sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right|.$$

We further examine the RHS and use our assumption that $x_1 \leq \dots \leq x_n$, which implies that for each i ,

$$\left| \sum_{j=1}^n (x_i - x_j) \right| = \max \left\{ \sum_{j=1}^i |x_i - x_j| - \sum_{j=i}^n |x_i - x_j|, \sum_{j=i}^n |x_i - x_j| - \sum_{j=1}^i |x_i - x_j| \right\}.$$

The idea will then be that because we have an ordering on x_1, \dots, x_n , there will be a transition index. In particular, there is some $l \in [n-1]$ such that

$$\left| \sum_{j=1}^n (x_i - x_j) \right| = \sum_{j=1}^i |x_i - x_j| - \sum_{j=i}^n |x_i - x_j|$$

for all $i > l$, and

$$\left| \sum_{j=1}^n (x_i - x_j) \right| = \sum_{j=i}^n |x_i - x_j| - \sum_{j=1}^i |x_i - x_j|$$

for all $i \leq l$. If we then have $l \leq k$, then,

$$\sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right| = \sum_{i=k}^n \left(\sum_{j=1}^i |x_i - x_j| - \sum_{j=i}^n |x_i - x_j| \right).$$

By cancellation, we get,

$$\sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right| = \sum_{i=k}^n \sum_{j=1}^k |x_i - x_j|.$$

Applying Fact A.1 gives,

$$2 \sum_{i=k}^n \sum_{j=1}^k |x_i - x_j| \geq \sum_{i=k}^n \sum_{j=1}^n |x_i - x_j| \geq \sum_{i=k}^n \sum_{j=1}^i |x_i - x_j|,$$

as desired. If $l > k$, then,

$$\begin{aligned} \sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right| = & \sum_{i=k}^l \left(\sum_{j=i}^n |x_i - x_j| - \sum_{j=1}^i |x_i - x_j| \right) + \sum_{i=l+1}^n \left(\sum_{j=1}^i |x_i - x_j| - \sum_{j=i}^n |x_i - x_j| \right). \end{aligned}$$

We will simply lower bound the first term in the sum by 0, and note that Fact A.1 (stated below) implies,

$$\sum_{i=k}^l \sum_{j=l}^n |x_i - x_j| \geq \sum_{i=k}^l \sum_{j=1}^i |x_i - x_j|.$$

Furthermore, by cancellation, we get,

$$\sum_{i=k}^n \left| \sum_{j=1}^n (x_i - x_j) \right| \geq \sum_{i=l+1}^n \sum_{j=1}^{l+1} |x_i - x_j|.$$

We then use the fact that,

$$\sum_{i=l+1}^n \sum_{j=1}^{l+1} |x_i - x_j| \geq \sum_{i=k}^l \sum_{j=1}^i |x_i - x_j|,$$

and

$$\sum_{i=l+1}^n \sum_{j=1}^{l+1} |x_i - x_j| \geq \sum_{i=l+1}^n \sum_{j=l}^i |x_i - x + j|,$$

from Fact A.1, to obtain:

$$3 \sum_{i=l+1}^n \sum_{j=1}^{l+1} |x_i - x_j| \geq \sum_{i=k}^l \sum_{j=1}^i |x_i - x_j|,$$

as desired. \square

Fact A.1. For any ordered values $x_1 \leq \dots \leq x_n$, and any $k \in [n-1]$ such that,

$$\sum_{j=1}^k |x_k - x_j| \leq \sum_{j=k+1}^n |x_k - x_j|,$$

then for any $i \leq k$, we must have,

$$\sum_{j=1}^k |x_i - x_j| \leq \sum_{j=k+1}^n |x_i - x_j|.$$

Proof. This follows from the fact that $x_1 \leq \dots \leq x_k$, so $\sum_{j=1}^k |x_i - x_j| \leq \sum_{j=1}^k |x_k - x_j|$ and $\sum_{j=k+1}^n |x_k - x_j| \leq \sum_{j=k+1}^n |x_i - x_j|$. \square

A.2 Omitted proofs from Section 6

In this section we prove some important facts about variance that were necessary for obtaining an efficient algorithm for variance. We first show the intuitive fact that if we want to decrease the variance most, we should remove the maximum or minimum value.

Proof of Fact 6.2. It suffices to show that $\mathbf{Var}[D - x_1] \leq \mathbf{Var}[D - x_i]$ if $x_i \leq \mu(D)$ and that $\mathbf{Var}[D - x_n] \leq \mathbf{Var}[D - x_i]$ if $x_i \geq \mu(D)$. We will show the first, and the second follows equivalently.

We again use the definition of variance stated as,

$$\mathbf{Var}[x_1, \dots, x_n] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j>i}^n (x_i - x_j)^2,$$

and by cancellation we see that showing $\mathbf{Var}[D - x_1] \leq \mathbf{Var}[D - x_i]$ is equivalent to,

$$\frac{1}{n^2} \sum_{j \neq 1} (x_i - x_j)^2 \leq \frac{1}{n^2} \sum_{j \neq i} (x_1 - x_j)^2,$$

which is also equivalent to showing,

$$\sum_{j \neq 1, i} (x_i - x_j)^2 \leq \sum_{j \neq 1, i} (x_1 - x_j)^2.$$

The proof then follows from the fact that this is a sum of least squares minimization for the vector $x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$, where we know that $x_1 \leq x_i$ and $\mu(D - x_1 - x_i) \geq \mu(D)$ because $x_1, x_i \leq \mu(D)$. \square

We now prove Fact 6.3 using the simple helper fact that if we add a data point and want to minimize the variance, then the added data point should be the mean of the remaining points.

Fact A.2. *Given any set $x_1, \dots, x_n \in \mathbb{R}$ with mean μ , then*

$$\arg \min_y \mathbf{Var} [y, x_1, \dots, x_n] = \mu.$$

Proof. By definition,

$$\mathbf{Var} [x_1, \dots, x_n] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j>i}^n (x_i - x_j)^2.$$

The problem we are considering fixes x_1, \dots, x_n and minimizes the variable y , so each term in the summation that does not include y can be ignored, and our minimization problem then reduces to,

$$\arg \min_y \mathbf{Var} [y, x_1, \dots, x_n] = \arg \min_y \sum_{i=1}^n (y - x_i)^2,$$

which is minimized when $y = \mu$. \square

We use this fact to lower bound the variance from adding one additional variable, and complete our proof of Fact 6.3.

Proof of Fact 6.3. We first upper bound the variance of all variables:

$$\min_y \mathbf{Var} [x_1, \dots, x_{n-1}, y] \leq \mathbf{Var} [x_1, \dots, x_n].$$

Fact A.2 implies that,

$$\min_y \mathbf{Var} [x_1, \dots, x_{n-1}, y] = \mathbf{Var} [x_1, \dots, x_{n-1}, \mu_{[1:n-1]}],$$

where $\mu_{[1:n-1]}$ is the mean of x_1, \dots, x_{n-1} . We then apply the definition of variance to get,

$$\mathbf{Var} [x_1, \dots, x_{n-1}, \mu_{[1:n-1]}] = \frac{1}{n} \sum_{i=1}^{n-1} (x_i - \mu_{[1:n-1]})^2.$$

Together, this implies that,

$$\min_y \mathbf{Var} [x_1, \dots, x_{n-1}, y] = \frac{n-1}{n} \mathbf{Var} [x_1, \dots, x_{n-1}],$$

which gives our desired result. \square

Finally, we also needed the following fact to reduce our running time to $O(n^2)$ for implementation of variance.

Proof of Fact 6.7. We utilize the definition of variance as,

$$\mathbf{Var}[x_1, \dots, x_n] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \frac{1}{2} (x_i - x_j)^2.$$

After cancellation from the scalars we have,

$$\begin{aligned} & \left(\frac{n-1}{n}\right)^2 \mathbf{Var}[D - x_a] + \left(\frac{n-1}{n}\right)^2 \mathbf{Var}[D - x_b] - \left(\frac{n-2}{n}\right)^2 \mathbf{Var}[D - x_a - x_b] + \frac{1}{n^2} (x_a - x_b)^2 \\ &= \frac{1}{n^2} \sum_{i \neq a} \sum_{j \neq a} \frac{1}{2} (x_i - x_j)^2 + \frac{1}{n^2} \sum_{i \neq b} \sum_{j \neq b} \frac{1}{2} (x_i - x_j)^2 - \frac{1}{n^2} \sum_{i \neq a, b} \sum_{j \neq a, b} \frac{1}{2} (x_i - x_j)^2 + \frac{1}{n^2} (x_a - x_b)^2. \end{aligned}$$

Separating out within the summation gives,

$$\frac{1}{n^2} \sum_{i \neq a, b} \sum_{j \neq a, b} \frac{1}{2} (x_i - x_j)^2 + \frac{1}{n^2} \sum_{i \neq a} (x_b - x_i)^2 + \frac{1}{n^2} \sum_{i \neq b} (x_a - x_i)^2 + \frac{1}{n^2} (x_a - x_b)^2,$$

which is equivalent to $\mathbf{Var}[D]$ as desired. □