Do I Get the Privacy I Need? Benchmarking Utility in Differential Privacy Libraries

GARRIDO, Gonzalo Munilla, NEAR, Joseph, AITSAM, Muhammad, HE, Warren, MATZUTT, Roman and MATTHES, Florian

Do I Get the Privacy I Need? Benchmarking Utility in Differential Privacy Libraries

Gonzalo Munilla Garrido*
TUM

Joseph P. Near
University of Vermont

Aitsam Muhammad
RWTH Aachen

Warren He
Oasis Labs

Roman Matzutt
RWTH Aachen

Florian Matthes
TUM

***Abstract*—An increasing number of open-source libraries promise to bring differential privacy to practice, even for non-experts. This paper studies five libraries that offer differentially private analytics: Google DP, SmartNoise, diffprivlib, diffpriv, and Chorus. We compare these libraries qualitatively (capabilities, features, and maturity) and quantitatively (utility and scalability) across four analytics queries (count, sum, mean, and variance) executed on synthetic and real-world datasets. We conclude that these libraries provide similar utility (except in some notable scenarios). However, there are significant differences in the features provided, and we find that no single library excels in all areas. Based on our results, we provide guidance for practitioners to help in choosing a suitable library, guidance for library designers to enhance their software, and guidance for researchers on open challenges in differential privacy tools for non-experts.**

***Index Terms*—Differential privacy, privacy-enhancing technology, scalability, recommendations.**

1. Introduction

In recent years, a confluence of trends drives academics and industry practitioners to research and invest in more powerful privacy-enhancing measures to protect people's privacy while leveraging their data. One of the drivers is the increase in costs for the institutions due to more frequent data breaches [1], e.g. loss of brand equity, customer turnover, or legal expenditure. Furthermore, white-hat academics have performed demonstration attacks on "de-identified" public data in different domains, effectively deprecating legacy privacy-enhancing methodologies. Notable examples of re-identification have taken place in genome sequencing [2], in the automotive, telecommunications, and entertainment industry [3, 4, 5], and in e-commerce [6]. Moreover, the promise of privacy-enhancing products and services can also bring benefits, such as avoiding price discrimination, allow the utilization of data across organizations, new business models, and develop less pervasive forms of social media that can prevent malicious social engineering [7], among others. In this context, differential privacy (DP) has received increasing attention because of its mathematical guarantees of privacy unique among anonymization techniques. Moreover, DP's inherent implementation complexity [8] has driven organizations and researchers to create libraries for practitioners to include DP in their stack.

In this paper, we examine a set of mainstream libraries through the lens of a benchmark. We consider open-source libraries that provide support for analytics queries, come from prominent institutions or researchers, and offer DP functionality that eases integration and usage. Thus, we consider the following libraries: Microsoft's SmartNoise [9], IBM's diffprivlib [10], Google-DP [11], Chorus [12, 13], and diffpriv [14, 15]. There exist other libraries that comply with the targeted qualities, e.g, the pioneering PINQ [16] and GUPT [17]; however, they are not maintained anymore.

At the core, these libraries aim to abstract DP to a level where *non-experts* can implement DP applications. As interest in differential privacy grows, research, governmental, and private institutions will gravitate towards these open-source libraries. Our work aims to help guide practitioners, library designers, and researchers in navigating the coming adoption of tools for DP.

Our Contributions. Specifically, this paper makes the following contributions:

- We conduct a comprehensive comparison and evaluation of five mainstream open-source DP libraries
- We provide *guidance for practitioners* to aid in selecting a specific library (§ 9)
- We provide *guidance for library designers* on how to make their software more useful (§ 10)
- We provide *guidance for researchers* on important open research challenges remaining in differential privacy tools for non-experts (§ 11)

We provide the source code [18] to reproduce our study and for practitioners to quickly implement further benchmarks.

Methodology. Our guidance is based on the answers to three research questions (RQ), described in Section 4. To answer these RQs, we compare these libraries both qualitatively (in terms of capabilities, features, and maturity) and quantitatively (in terms of scalability and utility through a set of principled experiments). Our benchmarks include both synthetic data (to explore the differences between libraries systematically) and real-world data (to confirm these results in a realistic setting). We compare the libraries across four query types: count, sum, mean,

*Corresponding author. Email: gonzalo.munilla-garrido@tum.de

and var.

Results & Key Findings. Our qualitative comparison (§ 6) demonstrates clear feature differences between libraries (see Table 2). First, libraries offer **different capabilities**: *e.g.* some target only analytics queries, while others provide machine learning capabilities as well. Second, libraries differ in **analyst support**: *e.g.* some calculate sensitivity automatically, while others require the analyst to provide it. Third, some libraries are designed for **direct use** by analysts, while others provide frameworks for **building applications**. In addition, libraries differ in their protections against **side channels** (such as floating-point vulnerabilities [19]) and support for **privacy budgeting** over multiple queries.

All of the libraries in our study provided **similar utility** for corresponding queries in our benchmarks, with some important exceptions (§ 7). We find that all five libraries offer similar performance characteristics, but that **none of the libraries** scales to truly massive datasets (§ 8).

Guidance for Practitioners. Our results suggest that the largest differences between libraries come in their support for analysts and protections against side-channels. We therefore advise practitioners to prioritize these factors when choosing a library. Libraries like diffprivlib provide the best support for data scientists, while Google-DP is designed for building new applications and provides strong protection against side-channels. Our complete guidance appears in Section 9.

Guidance for Library Designers. The results of our study indicate that library designers have generally done a good job implementing basic mechanisms and providing sufficient performance for small-scale analytics. We suggest that library designers prioritize support for the analyst, protections against side-channels, and the addition of "simple" mechanisms that can provide good real-world performance (like the Geometric mechanism). We also note the danger of implementation bugs in these libraries, and support the use of tools like Google-DP's stochastic tester to ensure correctness. Our complete guidance appears in Section 10.

Guidance for Researchers. Research in differential privacy has historically focused on developing mechanisms that improve utility. Our study suggests that the increasing adoption of differential privacy opens important new avenues for researchers in this area. In particular, practitioners need better tools for understanding **how much utility to expect** and **how to improve utility**. They also need help understanding the **non-privacy implications** of each mechanism, such as output consistency. Finally, as differential privacy sees increasing adoption, tools for **privacy budgeting** become even more important. Our complete guidance appears in Section 11.

The rest of this paper is organized as follows. We discuss related work in Section 2 and give background on differential privacy in Section 3. We describe our research questions and methodology in Section 4 and the datasets used in our study in Section 5. The results of our study appear in Section 6–8 and our guidance appears in Sections 9–11. Finally, we discuss our conclusion in Section 12.

2. Related work

DP is a powerful concept that causes ever-increasing interest among privacy experts and is currently an active field of research in academia and industry. Ever since the introduction of DP by Dwork [20], there has been abundant research conducted on the theoretical aspects of DP, addressing questions such as choosing the correct amount of noise and the appropriate values for DP parameters [21, 22, 23], as well as on the practical application of DP, for instance, in data mining [24], data publication and analysis [25], and deep learning [26].

Consequently, the expansion of the research and application of DP in recent years prompted researchers to perform systematic reviews and comparative studies of the work conducted in the field. Xiong et al. [27] and Yang et al. [28] present comprehensive surveys on local DP algorithms and their applications, providing a source of reference for different privacy-related scenarios, as well as identifying research gaps and possible directions for future research. Hassan et al. [29] conducted a detailed survey on the implementation of various DP techniques in cyber-physical systems, such as energy, transportation, healthcare, and industrial Internet of things. The authors covered all dimensions and aspects of implementing DP in these domains and discussed related issues and challenges. Furthermore, motivated by the argument that even aggregated data such as histograms can result in privacy leakage, Nelson and Reuben [30] conducted a systematic literature review, a qualitative analysis of 27 papers that address the application of DP for histogram and synthetic data. The authors identified trends in the field and explained a crucial issue in adopting DP to tackle the privacy-utility trade-off.

Aside from these qualitative surveys, there exist quantitative comparisons in the use of DP in range queries [31], geo-spatial data [32], spatial decomposition [33], data mining [24], and a test framework of DP [34]. However, the work most closely related to ours was conducted by Hay et al. [35] in 2016. In their paper, the authors propose a principled framework called DPBench to evaluate 1- and 2-dimensional range queries. However, none of the extant literature benchmarks the mainstream open-source DP libraries towards which non-experts gravitate; with this study, we fill this research gap.

3. Differential Privacy

Traditional privacy protection methods are vulnerable to auxiliary information attacks against sensitive data analysis public releases [2, 3, 4, 5, 6]. On the other hand, DP, introduced in 2006 by Dwork et al. [20], addresses the traditional techniques' limitations by mathematically formalizing a differential guarantee of privacy agnostic to auxiliary information. DP maintains this promise by ensuring that an informed adversary analyzing a query output cannot determine whether an individual's data was used to compute such output.

DP ensures that outcomes are similarly likely, *with* or *without* the data contributed by a particular individual. The similarity of outcomes is parameterized by the parameter ε, which tunes the strength of the privacy guarantee (a lower ε leads to better privacy). We define a state of

privacy as the prevention of the re-identification of an individual [36], whose protection is adjusted by ε. DP is formally described in Definition 1, which is based on [37]:

Definition 1 (*(ε,δ)-Differential Privacy*). A randomized algorithm $\mathcal{M}$ is (ε,δ)-differentially private if for any two datasets $\mathcal{D}$ and $\mathcal{D}'$ differing on at most one element, and any set of possible outputs $\mathcal{S} \in Range(\mathcal{M})$:

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \times \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta.$$

For $\delta = 0$, Definition 1 is considered *pure DP*. The weaker guarantee when $\delta > 0$ is called *approximate DP*. This relaxed form of DP lowers the privacy of the individuals in exchange for utility [38]. Additionally, one may choose how $\mathcal{D}$ and $\mathcal{D}'$ differ in one individual, leading to two possibilities: *bounded* or *unbounded* DP [39].

Mechanisms ensure DP by adding carefully-chosen random noise, typically to the output of a deterministic function. The functions used in this study are based on the analytics queries: count, sum, mean, and var. The scale of the noise used is based on the deterministic function's ℓ_1*-sensitivity* (*global sensitivity*) [38]:

Definition 2 (ℓ_1-sensitivity). The ℓ_1-sensitivity of an algorithm $\mathcal{W} : \mathbb{R}^m \rightarrow \mathbb{R}^n$, executed over datasets $\mathcal{D}$, $\mathcal{D}' \in \mathbb{R}^k$ at a Hamming distance of $d_h(\mathcal{D}, \mathcal{D}') = 1$, is:

$$\Delta f = \max_{\substack{\mathcal{D},\mathcal{D}' \in \mathbb{R}^k \\ d_h(\mathcal{D},\mathcal{D}')=1}} \|\mathcal{W}(\mathcal{D}) - \mathcal{W}(\mathcal{D}')\|_1.$$

There are multiple implementations of mechanisms complying with Definition 1 and 2. The most commonly known are the Laplace mechanism [40] and the Gaussian mechanism [38] for numerical data, and the Exponential mechanism [41] for categorical and numerical data. The Laplace and exponential mechanisms ensure pure DP, while the Gaussian mechanism requires approximate DP. In this paper, we benchmark algorithms derived from the Laplace mechanism [40]:

Definition 3 (Laplace mechanism). For an algorithm $\mathcal{W}$ executed over a dataset $\mathcal{D}$, its differentially private version $\mathcal{M}$ adds Laplace noise: $\mathcal{M}(\mathcal{D}) = \mathcal{W}(\mathcal{D}) + Lap(x|\mu, b)$, with center $\mu = 0$ and scale $b = \frac{\Delta f}{\varepsilon}$.

One may observe that the lower the ε, the larger the standard deviation (std) of the noise, improving privacy (but reducing utility). The noise magnitude is independent of the number of records in a dataset (dataset size), so analyzing more data yields better relative utility.

DP algorithms obey *sequential composition* [38], i.e. if a randomized query $\mathcal{M}$ is executed n times over $\mathcal{D}$ with ε_i, the total ε employed is equal to $\sum \varepsilon_i$, which is the consumed *privacy budget*. Because an adversary can use the n outputs to average out the noise because it is centered around 0, some libraries implement privacy budget trackers to help practitioners preventing these attacks. These trackers monitor the consumed budget and block further queries once the budget is consumed.

A function's sensitivity (Δf) depends on the function itself, and is sometimes difficult to calculate. Counting queries are easy: they have a sensitivity of 1. Other queries are more difficult: for example, the sensitivity of a summation query depends on the minimum and maximum possible values being summed (which themselves need to be kept private!). Thus, practitioners either set estimates without looking at the data, or employ libraries offering *private sensitivity estimation*, which consumes privacy budget.

Another practical consideration of DP algorithms is their *output consistency*, e.g., a var query should not output values ≤ 0, counts should not be decimal values, or the outputs of a DP count and sum from a library should yield similar values to the library's DP mean. Lastly, it is important to consider *side-channels* that may leak more information than the mathematical definition of the mechanism (like the *floating-point vulnerabilities* [19]).

Additional challenges of deploying DP include choosing ε and tracking budgets across different systems [42]. Despite its flaws, DP holds a set of advantages. Primarily, DP allows to mathematically limit the privacy loss of an analysis on sensitive data [38], and practitioners can adapt DP to different use cases, e.g., deep learning [43]. Moreover, the community of researchers continuously tackles the flaws of DP, such as solving its floating-point vulnerability with the Snapping mechanism [19] and improving this mechanism's utility (SmartNoise, Google DP). Overall, because of its unique features, DP may become the de facto standard of privacy in the near future.

4. Methodology

4.1. Research questions

To provide practitioners, library designers, and researchers with a comprehensive picture of the qualities and performance of the five selected open-source libraries, we aim to answer the following research questions (RQ):

RQ1 *How do available open-source libraries compare in terms of functionality?* To answer this RQ, we conduct a qualitative comparison in Section 6.

RQ2 *How do available open-source libraries compare regarding utility in the count, sum, mean, and var queries?* In answer to this RQ, we perform a utility benchmark of the analytics queries in Section 7.

RQ3 *Which libraries perform best regarding execution time and memory consumption as the dataset size increases?* Section 8 tackles this RQ by executing a scalability benchmark.

After answering these RQs, we extracted a set of recommendations for practitioners (§ 9) and library designers (§ 10), and guidance for researchers (§ 11).

4.2. Principles for evaluation

Our evaluation focuses on four commonly-used analytics queries: count, sum, mean, and var. We base our evaluation principles on DPBench, developed by Hay et al. [35]; most of the other literature focuses on comparing algorithms without formalizing the process [24, 31, 32, 33]. We formulate our comparison Principles in Table 1.

Principles from I to V are necessary to lay the basis for a holistic comparison of the libraries. Specifically, we employed real datasets and created synthetic datasets (I) to make our comparisons comprehensive and reproducible. We employed the skew-normal distribution to add diversity to our synthetic data and to provide a reasonable model of realistic data distributions; consequently, we

TABLE 1: Principles of comparison adopted from or inspired by DPBench [35].

	Principle	Description	Implementation
(I)	Datasets of synthetic and real nature	*The algorithms inputs should comprise synthetic and real data.*	We employ synthetic data to surface differences between algorithms and real data to confirm these results.
(II)	Dataset size diversity	*Execute algorithms on datasets of varying numbers of records.*	The synthetic datasets contain 1000, 10000, or 100000 records.
(III)	Shape diversity	*Execute algorithms on datasets of varying record distribution over the domain.*	To make synthetic datasets more diverse, we set the *skew* parameter of Scipy's skew-normal noise generator [44, 45] to the values 0, 5, 50 (The location parameter was set to 0).
(IV)	Spread diversity	*Execute algorithms on datasets of varying spread.*	Likewise, aiming to produce more diverse datasets, we tuned the synthetic datasets spread with the *scale* parameter of Scipy's skew-normal noise generator [44, 45] with the values 50, 250, and 500.
(V)	ε diversity	*Execute algorithms under different ε values.*	We perform experiments for 73 ε values from 0.01 to 100.
(VI)	Measurement of expectation, variability, and bias	*The benchmarks results should register accuracy, precision, and bias to measure utility.*	The output of our benchmark measures the outputs' sample mean of the relative error (accuracy) and sample std of the absolute scaled error (precision). Measuring accuracy at values of ε up to 100 can reveal an algorithm's bias.
(VII)	Comparable results	*The results should be comparable to other algorithms beyond the ones compared in the benchmark.*	We measure the relative error so that other practitioners can compare their algorithms.
(VIII)	Avoid extreme input settings	*The algorithms inputs should not lead to edge-case comparisons.*	We do not input extreme values such as scales of a handful of records.

varied the number of records (II), how these records are distributed over the domain by adjusting the skew parameter (III), and the spread of the datasets by changing the scale parameter (IV). Our synthetic datasets contain continuous data, since the benchmarked algorithms work equally well for continuous and discrete data (and are not impacted by domain size). Consecutively, to ensure ε diversity (V), we ran experiments for 73 ε values from 0.01 to 100 at logarithmic steps, except for ε values between 0.01 to 0.5, for which we employed steps of 0.01. We chose this fine granularity to reveal the behavior at small ε values because practitioners mainly adhere to this range [8], as privacy is conserved most.

Principle VI deals with the fact that the outputs of the algorithms are r.v.'s. Thus, to measure the results' utility, we perform 500 experiments for each ε (500 x $73 = 36500$ experiments) to have a large enough sample for each library and input dataset. Moreover, we set the outputs of our benchmark to be the relative error and the absolute scaled error w.r.t the dataset scale. Consequently, the values we compare between libraries are the sample std of the absolute scaled error (SASE), and the sample mean of the relative error (MRE), providing two summary values per ε. The baselines for the error calculations are the deterministic outputs of the analytics queries. We choose the MRE because it is familiar to practitioners and widely used in utility comparisons in the extant literature [35], and allows to compare datasets of different scales; it is a measure of the accuracy of the random query. In contrast, the SASE serves to understand the variability of the outputs across datasets of different scales, i.e., it measures the precision of the random query. Without scaling the error, the SASE would remain the same across scales; however, in practicality, an error spread of equal magnitude imbues more uncertainty in small datasets than in large ones. Scaling makes the comparison across dataset scales fair.

The MRE is informative for "risk-neutral" practitioners, while the SASE is for the "risk-averse" because the metric indicates the presence of outliers in the outputs [35]. For each ε, we consider the libraries with smaller SASE and MRE to perform better. Furthermore, measuring the MRE for large values of ε (up to 100) will reveal the presence of an algorithm's bias. Because the MRE's underlying value is the relative error, we enable practitioners to use our results as a baseline for comparing other DP algorithms (VII). Finally, we did not employ outlier values for the generation of the synthetic datasets to avoid comparisons that may not be equivalent to reality (VIII).

For the scalability benchmark, we measure execution time (t) and memory consumption (m). We perform this benchmark by obtaining t and m from running an analytics query at varying dataset scales from 10 to 10 million data points with a fixed skewness and scale. We limit the measurements to the query execution itself, not considering: pre-processing steps, dispatching, and receiving the query result. This experimental setting is preferred because these three steps may vary between practitioners' system architectures.

5. Datasets overview

5.1. Synthetic datasets

We chose three values for each of the characteristics proposed by principles II (*dataset size diversity*), III (*shape diversity*), and IV (*spread diversity*) following our implementation guidelines of Table 1, obtaining 27 datasets[1] (see Fig. 1).

Some queries require knowledge about the range of possible values in the dataset. Sometimes, domain knowledge can be used to define this range (e.g. human ages are typically between 0 and 100), but in other cases (like our synthetic data), this process is more challenging. Google DP provides a DP mechanism for estimating the range of a dataset, but the other libraries require the analyst to specify the range (including Google DP's Python wrapper). To ensure consistency in our synthetic data experiments, we use the *actual* maximum and minimum values of the dataset when such a range is required.

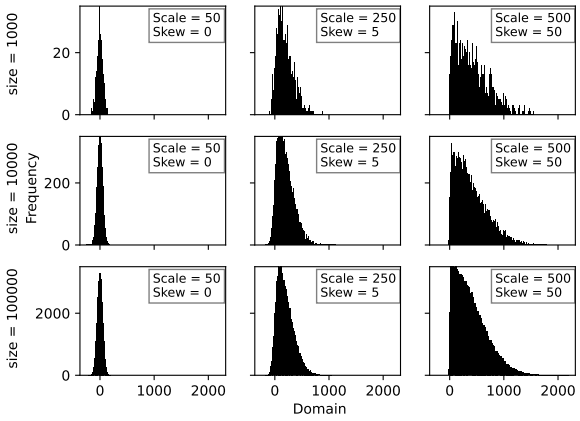

Figure 1: 9 of the 27 normally distributed synthetic dataset histograms.

More formally, we represent each D_i as a data vector $\mathbf{x}$, which contains a list of *float* values (within the set of real numbers), that varies in size, shape, and scale (based on Principles II to IV) across different datasets D_i, for $i \in [1, 27]$. For example, dataset D_1 was generated with a skewness of 0, a scale of 50, and 1000 data points. The *workload*, $\mathbf{W}$, is a set of four one-dimensional functions (analytics queries, $\mathcal{W} \in \mathbf{W}$) executed over $\mathbf{x}$. Applying the $\mathcal{W}$ to $\mathbf{x}$ yields the vector of deterministic results $\mathbf{y} = \mathcal{W}(\mathbf{x})$. The libraries transform an analytics query $\mathcal{W}$ from the workload $\mathbf{W}$ into a DP algorithm $\mathcal{M}$ by adding noise to the result of $\mathcal{W}(\cdot)$, i.e., the noise generates randomized analytics queries. The noisy result is $\hat{\mathbf{y}} = \mathcal{M}(\mathbf{x}) = \mathcal{W}(\mathbf{x}) + Noise$, and we use the L_1 norm as the error of $\mathcal{M}$, $\|\mathcal{W}(\mathbf{x}) - \hat{\mathbf{y}}\|_1$, scaled by the deterministic value $\mathcal{W}(\mathbf{x})$ for the relative error, and by the cardinality of the dataset $|\mathcal{D}_i|$ for the absolute scaled error. The libraries' goal is to report the approximate results of the queries in $\mathbf{W}$ on the private datasets in $\mathcal{D}$ while incurring minimal error.

1. The 27 datasets were generated using skew-normal noise with the Python package Scipy [44, 45].

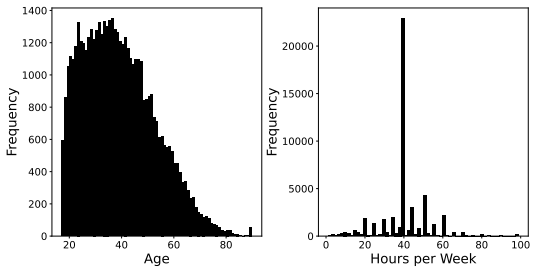

Figure 2: 1984 USA Census dataset attributes of *age* and worked *hours per week* containing 48842 individuals.

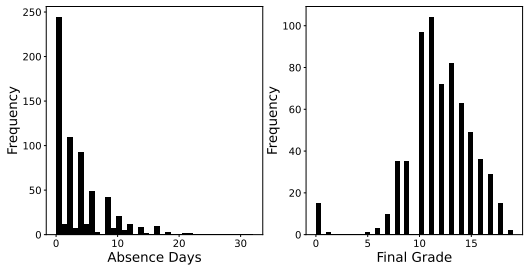

Figure 3: Portuguese education dataset attributes of *absence days* and *final grade* containing 649 students.

Lastly, to test memory consumption and execution time, we carried out experiments on datasets of fixed *shape* and *spread* (generated with a skewness of 5 and a scale of 250) but of varying *dataset size* from 10 to 10 million data points.

5.2. Real-World datasets

Benchmarking analytics queries solely on synthetic data is not enough, as practitioners will ultimately execute the queries over real-world datasets. Therefore, we must also select publicly available datasets for the benchmark. Consequently, we have chosen two publicly available demographic datasets from different sources and contexts: the 1994 USA census [46, 47] with 48842 individuals, and a Portuguese education dataset [48, 49] with 649 students. To increase the diversity of inputs of the benchmark, we select two sensitive numeric attributes from each of the two datasets: *age* and *hours worked per week* from the census dataset, and *absence days* and *final exam grade* from the education dataset; histograms of these attributes may be observed in Fig. 2 and Fig. 3. Unlike for the synthetic datasets, to set the range bounds for the sensitivity calculation, we selected values based on the domain knowledge of the real-world datasets' attributes, e.g., for the *age* attribute, we selected a lower bound and an upper bound of 0 and a 100 years, respectively. Lastly, we formally define the real-world datasets and the workload equally to the synthetic datasets.

6. Qualitative comparison (RQ1)

Table 2 provides an overview of the main characteristics of the benchmarked libraries. We detail these characteristics below.

diffprivlib. Developed in Python under MIT license, IBM's diffprivlib [10] is a general-purpose library for data

TABLE 2: Qualitative overview of the selected five open-source libraries.

Features	**diffprivlib**	**SmartNoise**	**Google-DP (PyDP)**	**diffpriv**	**Chorus**
Contributor	IBM	Microsoft	Google (OpenMined)	B. Rubinstein et al.	J. P. Near et al.
Programming Language	Python	Python wrapper over Rust runtime	Google-DP: C++, Java, Go (PyDP: Python wrapper over C++)	R	Scala
Primary use	Data science facing operations (notebooks)	Data science facing operations (notebooks), and large-scale systems	Google-DP: Production-ready applications (PyDP: Data science)	Data science	Large-scale systems
Unique value proposition	Numerous machine learning algorithms, and DP mechanisms for experimentation	Blend of data science and operations	Google-DP: Deployment of applications, e.g., in mobile phones (PyDP: Data science)	Flexibility for data scientists: User-defined functions and empirical calculation of sensitivity	Scalability via cooperation with existing databases; extensibility
License	MIT	MIT	Apache-2.0	MIT	MIT
Benchmarked version	0.4.0	0.2.2	1.0.1	0.4.2	0.1.3
Functional features					
Mechanisms	Laplace, Gaussian, Exponential, Geometric, Staircase, Binary, Bingham, Vector, and Uniform	Laplace, Gaussian, Exponential, Geometric, and Snapping	Google-DP: Laplace, Gaussian, Exponential, and Snapping (PyDP: Laplace)	Laplace, Gaussian, and Exponential	Laplace, Gaussian, Noisy Max, FLEX, SVT, Sample & Aggregate
Analytics queries	Count, Sum, Mean, Var, Std, and Histogram	Count, Sum, Mean, Var, Covar, Histogram, Quantile, Maximum, Minimum, Median, and Raw Moment	Count, Sum, Mean, Var, Std, Maximum, Minimum, and Median	Any, provided the sensitivity sampler. However, none are built-in	Count, Sum, Mean, Histogram
DP definition for analytics queries	Bounded	User defined	Unbounded	User defined	Unbounded (Most mechanisms); bounded (FLEX mechanism)
Privacy budget accounting	Available	Available	Available	N/A	Available
Sensitivity calculation (private sensitivity calculation)	Available (N/A)	Available (N/A)	Available (Available)	Only with the sampler (N/A)	Available (N/A)
Floating-point vulnerability protection	N/A	Snapping mechanism for the Laplacian distribution	Snapping mechanism for the Laplacian and the Gaussian distributions	N/A	N/A
Differentially private machine learning algorithms	K-means, Linear and Logistic Regression, Naive Bayes, and PCA (tools: Standard scaler)	Linear Regression	N/A	Bernstein (built-in). Provided the sensitivity sampler, user-defined: SVMs, Bayesian inference, feature selection, among others [15]	N/A

scientists. diffprivlib differentiates itself by including a plethora of DP mechanisms that go beyond the Laplacian, the Gaussian, and the Exponential, by additionally including the Geometric [50], the Vector [51], the Staircase [52], among others (see Table 5). Additionally, diffprivlib offers some of these mechanisms in various forms, namely their truncated and the bounded form [53]. However, not having floating-point safety for a differentially private mechanism is a vulnerability. Altogether, the unique value proposition of diffprivlib is the comprehensive catalog of mechanisms, analytics queries, and DP machine learning algorithms, which run like Sklearn classifiers with the option to track the privacy budget.

SmartNoise. Implemented under MIT license in Rust (runtime) and bound to a Python wrapper, Microsoft in collaboration with the OpenDP initiative led by Harvard IQSS and SEAS provide SmartNoise [9] as an open-source library of DP mechanisms for the release of statistics, APIs for defining DP analysis, a validator to perform privacy budget accounting, and metadata concerning the utility of the outputs. Furthermore, SmartNoise offers floating-point safety, allows the user to choose from bounded and unbounded DP definitions, and offers the most extensive set of analytics queries. However, SmartNoise lacks DP machine learning implementations, and it is not easy to work with SmartNoise when there are multiple datasets involved, as one must follow a specific design pattern instead of a more familiar one using, e.g., Sklearn. Despite its data science limitations, SmartNoise compensates by offering an architecture designed towards large-scale systems.

Google DP. Google's library [11] under Apache-2.0 license offers a suite of DP analytics queries. The Python wrapper (PyDP) around C++ built by OpenMined makes Google DP accessible to data scientists. Furthermore, this library provides a layer for non-experts based on Apache Beam, while an expert may use the DP building blocks directly. Google DP can automatically approximate bounds for the analytical sensitivity formulas if none were input, enables privacy budget tracking, and checks whether any DP guarantee has been broken in the analysis through a stochastic tester. Moreover, developers may deploy operations-ready applications with the underlying C++ codebase. However, Google DP does not contain any machine learning algorithms.

diffpriv. diffpriv [14, 15] is a package developed in R, under MIT license, that enables data scientists to execute user-defined functions in a DP manner, such as analytics queries or model fit. Furthermore, diffpriv's sampler can empirically calculate the sensitivity of a user-defined function under the bounded definition of DP [15, 54], so that non-experts do not need to calculate sensitivity analytically, which may be arduous for machine learning algorithms. Rubinstein et al.'s sensitivity sampler ensures that DP holds with high probability [15], i.e., one assures *random* DP, but not pure DP. We, however, decided to use the sensitivity sampler for the analytics queries because we target non-experts. Additionally, we also ran another set of the experiments providing the same analytical sensitivity formulas for bounded DP that SmartNoise (Microsoft) has used from Harvard's Privacy Tools Project [55] (coincidentally the same as in diffprivlib). However, there are also shortcomings: the empirical calculation of sensitivity is computationally expensive even for simple queries, and diffpriv does not offer floating-point safety or a privacy budget accountant.

Chorus. Chorus [12, 13] distributed as a Scala library is a research system developed specifically to explore the use of DP at scale—for example, in production datasets at tech companies, which often contain billions or trillions of records and do not fit in memory. To query data at this scale, organizations often build and deploy extensive infrastructure. Chorus aims to work in *cooperation* with the existing infrastructure by using an existing SQL database to perform the "heavy lifting" of executing queries on large-scale datasets. Chorus provides three components: a query analysis framework (e.g., determining the sensitivity of an analyst-specified query), a query rewriting framework (e.g., modifying a query to perform clipping before executing it), and a privacy budget accountant. The resulting DP mechanisms aim to maintain roughly the same scalability properties as the underlying database infrastructure. Note that Chorus is a research framework and does not provide an out-of-the-box system ready for deployment; moreover, it has fewer built-in mechanisms and is less ready for production use. However, Chorus is unique in its ability to scale to large datasets by cooperating with an existing high-performance database.

7. Utility benchmark (RQ2)

7.1. Setup

We conducted the experiments running Ubuntu 20.04.1 LTS [57] on one server (Intel Xeon E5-2650 v2 16-core CPU, 32 GiB of memory). We used Python 3.8.5, R 4.0.3, and Scala in version 2.10, to run the latest stable versions of each considered library at the time of the benchmark. Using the available cores in parallel, we independently conducted each experiment 500 times per ε for the utility experiments, and 5 times per dataset size in the scalability benchmark of Section 8. To improve plot readability, in the body of this paper we depict the plots for $\varepsilon \leq 10$ because utility for $\varepsilon > 10$ was predominantly equal among libraries (see Appendix). Algorithm 1 of the Appendix depicts the workflow of the experiments concerning analytics queries executed on the 27 synthetic datasets and the two real-world datasets, jointly represented by $\mathcal{D}$.

7.2. Benchmarked algorithms

We benchmarked the default mechanisms because we target practitioners without in-depth knowledge of DP. All the default mechanisms from Table 3 are derived from the Laplace mechanism, including the Snapping mechanism [19] and the Geometric mechanism [50]. The Snapping mechanism protects against the floating-point vulnerability by executing a succession of steps such as sampling from a uniform distribution, clamping, and rounding to the closest multiple of a power of 2 [19]. Furthermore, the Snapping implementations from SmartNoise and Google DP differ, and add less noise than the original work of Mironov [19]. On the other hand, the Geometric mechanism is a discrete variant of the Laplace mechanism

TABLE 3: Default mechanisms and DP definitions (Bounded or unbounded) of the libraries used for each query in our benchmark. For more details, see Appendix Table 5.

Library	Count	Sum	Mean	Var	Floating-point safety
diffprivlib (IBM)	Uses the sum query to add the count (*ints*) of non-zero values: Geometric Truncated	Laplace Truncated for *floats* and Geometric Truncated for *ints*	Laplace Truncated	Laplace Bounded Domain	N/A
	Unbounded DP	Bounded DP	Bounded DP	Bounded DP	
SmartNoise (Microsoft)	Pure Geometric	Snapping Laplace for *floats*, and Pure Geometric for *ints*	Uses the sum and count query mechanisms: Snapping Laplace, and Pure Geometric, respectively	Uses the mean query to compute: $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. Therefore, in turn, uses Snapping Laplace for the sum and Pure Geometric for the count	Default
	Unbounded DP	Bounded DP	Bounded DP	Bounded DP	
Google-DP	Snapping Laplace	Snapping Laplace	Noisy average with normalization [56]: Uses the sum and the count query mechanisms, i.e. Snapping Laplace	Uses the mean query to compute: $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. Means computed according to [56], therefore, in turn, var uses the count and sum queries: Snapping Laplace	Only option
	Unbounded DP	Unounded DP	Unbounded DP	Unbounded DP	
diffpriv (Rubinstein et al.)	Pure Laplace	Pure Laplace	Pure Laplace	Pure Laplace	N/A
	Unbounded DP	Bounded DP	Bounded DP	Bounded DP	
Chorus (Johnson et al.)	Pure Laplace	Pure Laplace	Uses the sum and count query mechanisms, i.e., Pure Laplace	N/A	N/A
	Unbounded DP	Unbounded DP	Unbounded DP	Unbounded DP	

that satisfies DP with equality and, therefore, produces tighter guarantees for integer-value outputs, resulting in higher accuracy. Note that the Geometric mechanism is inherently invulnerable to a floating-point attack because its distribution is supported on the integers. Aside from these variants, library developers (namely diffprivlib's) have truncated or bounded the domain of the Laplace distribution to preserve output consistency while holding under DP, e.g., preventing counts < 0 or vars ≤ 0.

Aside from the mechanisms, there are different ways to implement a query (see Table 3). diffprivlib implements specific DP algorithms for each query except for the count, which reuses the sum query. Google, Microsoft, and Chorus use the count and the sum queries as building blocks for the mean and var queries (with $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$); however, Chorus does not implement the var query. Lastly, diffpriv enables all queries with the Laplace mechanism.

7.3. Experiments on synthetic datasets

In this Section, we introduce the experiments' results, namely the behaviour of the dependent variables presented in Section 4, i.e., the sample mean of the relative error (MRE—accuracy) and the sample std of the absolute scaled error (SASE—precision), w.r.t. the independent variables (dataset size, skewness, and scale), from which we derive the recommendations of Sections 9 and 10.

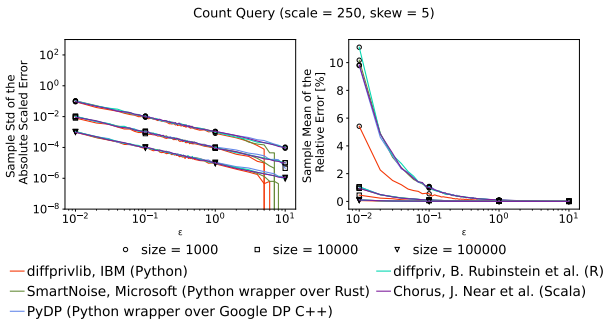

Figure 4: SASE and MRE of the count query for experiments with synthetic datasets.

Count. The count of records in a dataset is only affected by the dataset size; thus, we dismiss the other independent variables, i.e., skewness and scale. The most

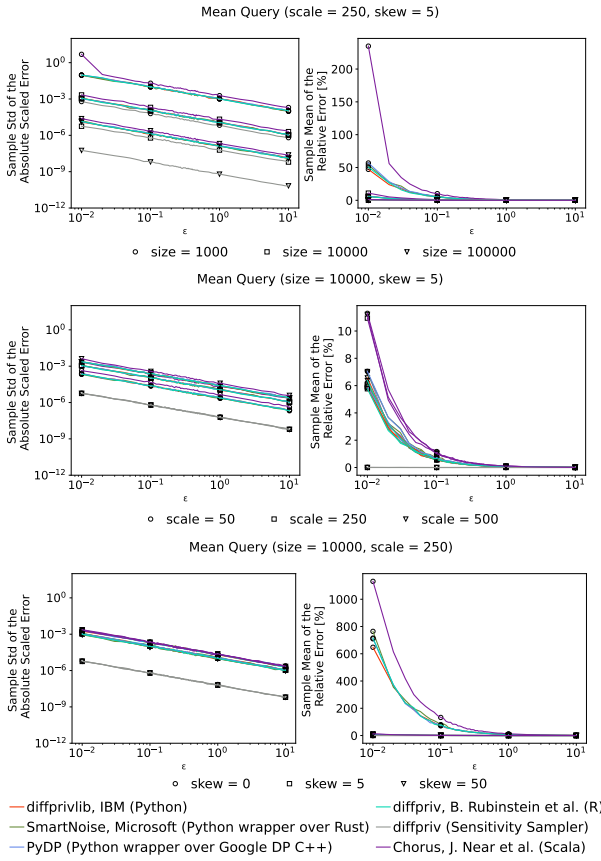

Figure 5: SASE and MRE of the mean query for experiments with synthetic datasets.

characteristic behavior in the left plot of Fig. 4, which measures the SASE, is the drops at around $\varepsilon = 10$ due to rounding. The noise added at such values of ε by diffprivlib, SmartNoise, and Google DP is small enough for the output rounding to the nearest integer to produce the same value consistently. However, diffpriv and Chorus do not round the output, which is why the curves continue in the same manner beyond $\varepsilon = 10$ (See Appendix). Overall, because the global sensitivity of the count query remains constant at the value of 1 for any dataset size, i.e., the DP noise distribution remains unchanged, the larger the dataset size, the lower the impact of the noise relative to the deterministic result (diminishing MRE), and the lower the impact of the noise spread on the absolute scaled error (diminishing SASE).

Mean. We selected the mean query to show the libraries' behavior across the three independent variables: dataset size, skewness, and scale[2]. Fig. 5 shows that the independent variables with the highest impact on the SASE are dataset size, followed by the scale, and lastly by the skewness. On the other hand, one can observe significant MRE values on the outputs for datasets generated with low skewness and small dataset size for $\varepsilon \leq 1$.

(i) Regarding dataset size, see Fig. 5 top plot. As dataset size decreases, the global sensitivity for the mean increases, which, in turn, increases the spread of the DP noise, resulting in higher SASE and MRE. (ii) Concerning the datasets' scale, see Fig. 5 middle plot. Larger scales lead to higher dataset spreads and global sensitivity, and therefore also higher SASE and MRE. (iii) Regarding skewness, see Fig. 5 bottom plot. While not as much as the scale parameter, the skewness parameter of the skew-normal also affects the spread of the distribution, from which we sampled the datasets. Thus, we observe diminishing MRE as skew increases with a fixed scale, and the underlying reason is the decrease of the datasets' spread. However, the impact in the SASE is not as notable.

2. Given the similarity in behaviour across queries w.r.t. the independent variables, to describe the dependencies between the variables in the most concise manner, we consider the sensitivity of the mean and not of the combination of the count and sum.

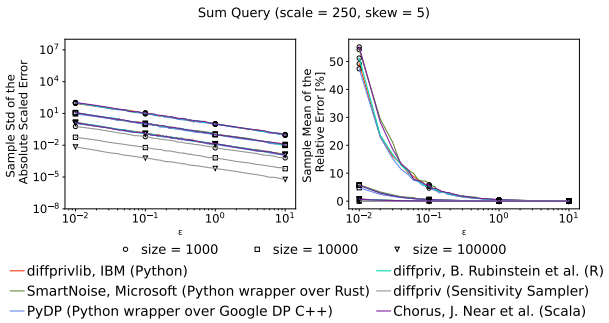

Figure 6: SASE and MRE of the sum query for experiments with synthetic datasets.

Sum. Overall, the behavior induced by variations of the three independent variables in the sum query is roughly equivalent to the behavior noted in the mean query (see Fig. 6). This similarity is the result of some libraries using the sum and count queries as building blocks for the mean query algorithm (Google DP, Smartnoise, and Chorus) or the same underlying mechanism (diffprivlib, diffpriv) (see Table 3). Aside from the behavior covered discussing the mean, these commonalities make the sum and mean queries display a corner-case behavior when the scale is large while the skewness is low. In these scenarios, the effect of the dataset's scale on the sensitivity could be significant enough to counter the effect of a larger dataset size, which would lower the relative error. Consequently, in such a context, the SASE results from a larger dataset are not necessarily better than with a smaller dataset; this is because there is room for more outliers in the larger dataset that increase the difference between the range clipped bounds, which, in turn, increases the global sensitivity (see the top plots in Table 6 of the Appendix). In these cases, the tool that provides better utility for the sum query for ε values less than 1 is Google DP's Snapping Laplace mechanism; this makes Google DP's sum query relatively better in terms of privacy protection and utility. Outside of these conditions, however, the libraries perform similarly.

Var. Equivalently to the mean, most of these libraries utilize the sum and the count as building blocks for the variance algorithm or the same mechanism (see Table 3). Thus, the overall behavior is also similar to the sum across the three independent variables (see Fig. 7), except in the case of diffprivlib, which uses a distinct algorithm for the var. Furthermore, note that some libraries offer the std query as a built-in function (see Table 2); nonetheless, their algorithms calculate the std by the square root of the var query output, leveraging the post-processing properties of DP. Prominently, the var is the query where diffpriv's

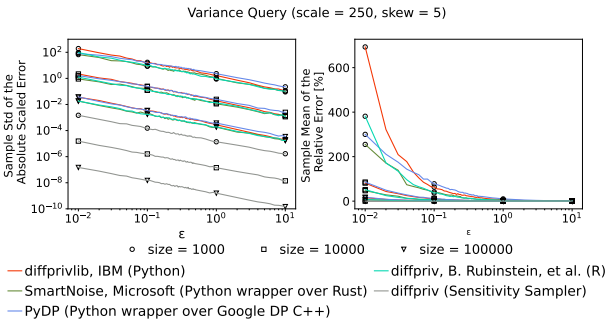

Figure 7: SASE and MRE of the var query for experiments with synthetic datasets.

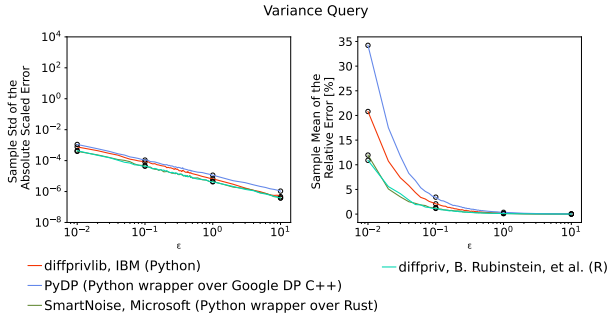

Figure 8: SASE and MRE of the mean of Age in the USA census dataset with 48842 data points.

sensitivity sampler has significantly outperformed the rest of the libraries. Lastly, note that Chorus has no var query functionality.

7.4. Experiments on real-world datasets

The utility results obtained with the real-world datasets show the same patterns across queries, e.g., in the var query of Fig. 8. Notably, from the results, we observe that for the USA census dataset with 48842 data points, the libraries provide similar values of SASE and MRE for the mean and the sum, except for Chorus. Moreover, the utility results of the Portuguese Education dataset with 649 data points suffer due to the sparsity over the domain of values in both attributes and the small size of the dataset, especially for the var query (see Tables 14 and 15 in the Appendix). Moreover, Fig. 9 depicts significant MRE values produced by diffprivlib in the count query.

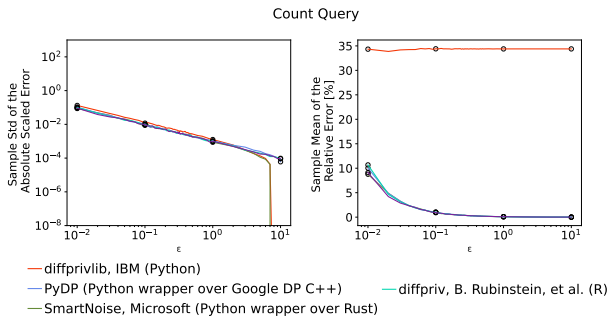

Figure 9: SASE and MRE of the count of Absences in the Portuguese Education Dataset with 649 data points.

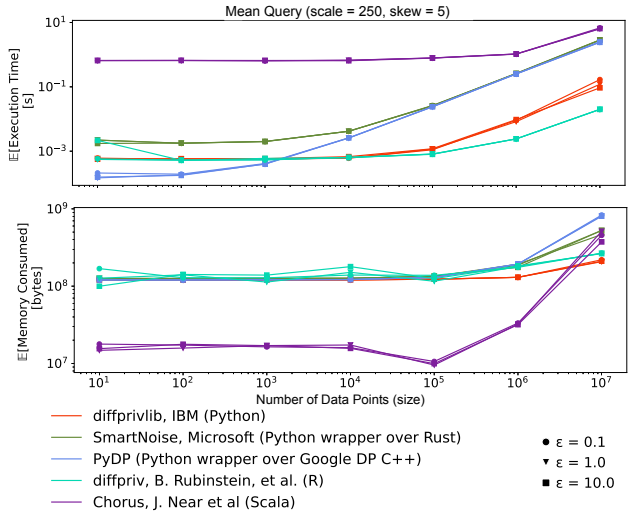

Figure 10: Libraries' execution time and memory consumption for experiments with synthetic datasets of varying dataset size.

8. Scalability Benchmark (RQ3)

We selected the mean query to discuss scalability; however, the behaviour remains uniform across queries (see Tables 10 and 11 of the Appendix). Note that we used different ε values as sanity check for the implementation, as the scalability should be independent to the sampled noise.

Based on the execution time analysis in Fig. 10, we conclude that Chorus scales several orders of magnitude worse than the rest of the libraries across all dataset sizes. Furthermore, while the rest of the libraries perform similar up to 10000 data points, all libraries tend to perform slower on larger datasets, specially the ones with a Python wrapper (Google DP and SmartNoise). Lastly, diffpriv scales best at the largest dataset size, followed by diffprivlib. Nonetheless, SmartNoise and Google DP might be faster without the wrappers; however, we have not conducted experiments in this direction. On the other hand, analyzing the memory consumption of Fig. 10, we conclude that Chorus outperforms the rest of the libraries for datasets of less than 10 million data points. Moreover, despite the low-level implementations in C++ of Google DP and Rust of SmartNoise under their Python wrappers, their overall memory consumption is equal to or worse than the rest of libraries.

9. Guidance for Practitioners

Do I get the privacy I need? *Yes*—our results suggest that the libraries we studied contain robust implementations of well-studied mechanisms, and they can generally be used to provide differential privacy. We see different noise levels across different libraries in our experiments, but not to an extent to suggest that some libraries would fail the DP criteria. Depending on the threat model of a particular deployment, however, it may be vital to consider side channels like floating-point protection (discussed in detail later in this section).

Which library should I choose? Based on our results,

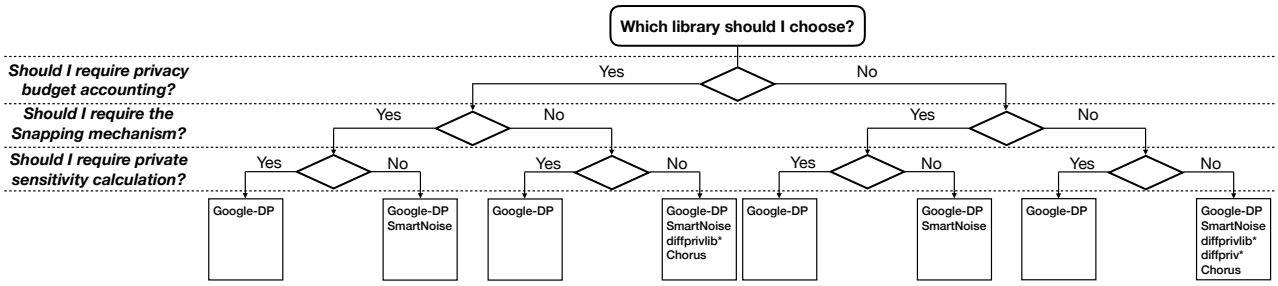

Figure 11: Decision tree for choosing DP libraries based on side channels.*Provide DP machine learning functionality

the libraries share enough similarities for practitioners to feel comfortable choosing any of the libraries. However, if the practitioner is an analyst or data scientist, we recommend diffprivlib primarily due to its output consistency, ease of use, and a wide variety of DP mechanisms and machine learning models. On the other hand, if the practitioner is a developer whose applications will expose data to a broader audience, we recommend using Google DP chiefly because it tackles more side-channel attacks than the rest of libraries. For practitioners concerned with side channels, we have compiled a decision tree presented in Fig. 11. For practitioners primarily concerned with utility and scalability, we recommend referencing Table 4. We detail additional considerations below.

RQ1. From our qualitative comparison, we extracted the following recommendations:

Integration. (i) For endeavors concerning exposing data to the public or in a world-facing application, we recommend the libraries that offer privacy budget tracking and protection against the floating-point vulnerability with the Snapping mechanism (Google DP and SmartNoise). (ii) Practitioners needing to integrate with existing large-scale query infrastructure can employ Chorus, Google DP, or SmartNoise; however, Chorus needs added code for deployment. (iii) On the other hand, in development, research, or data science, we recommend the mechanisms with the best utility for the desired level of privacy while being mindful about side channels that can influence privacy in ways not captured by ε.

Ease of use and available mechanisms. Notably, diffprivlib offers easy-to-use syntax, many mechanisms, and some machine learning features. Moreover, diffprivlib and diffpriv are designed for data scientists: the features of these libraries are integrated into existing Python and R syntax, e.g., diffprivlib can run DP classifiers with SKlearn, unlike the C++ (and its Python wrapper PyDP) or Scala implementations of Google DP and Chorus, respectively. On the other hand, Google DP, Chorus, and SmartNoise are more fitting for developing new applications due to their architectural components.

Side channels. (i) To address floating-point vulnerabilities, practitioners can employ the Snapping mechanism from SmartNoise or GoogleDP, and the Geometric mechanism from diffprivlib for counts or sums of integers. (ii) Additionally, budget tracking is necessary to avoid reverse-engineering the outputs; the only library that does not include budget tracking is diffpriv. (iii) Lastly, libraries should include features to help practitioners find the clipping bounds of a dataset's value range without looking at the data for the sensitivity calculation. However, Google DP is the only library that includes private sensitivity calculation for unbounded datasets.

TABLE 4: Library recommendations based on the quantitative benchmark of the selected five open-source libraries.

Utility benchmark	
Count	diffprivlib
Sum	Google-DP
Mean	Not Chorus
Var	diffprivlib
Scalability benchmark	
Execution time	diffpriv
Memory consumption	Chorus

RQ2. Based on the utility benchmark, we provide the library recommendations of Table 4 and highlight the following points:

Count. Based on the experiments of fig. 4, (i) for small and medium-sized datasets (1000 to 10000 records), diffprivlib performs notably better in terms of accuracy. (ii) The Geometric mechanism (SmartNoise) performs similar to the Laplace mechanism in accuracy, while the Geometric Truncated (diffprivlib) brings a substantial improvement. (iii) However, diffprivlib's implementation of the count is a conditional sum of non-zero values. The drawbacks of this implementation is noticeable in the experiments of Fig. 9, where 156 records have zero values, which are omitted and make the true count from diffprivlib's perspective equal to 493 instead of 649, generating significant MRE values. (iv) Lastly, diffpriv and Chorus do not round the querys output.

Mean. Looking at Fig. 5 as a whole, (i) Chorus performs slightly worse regarding precision and significantly worse regarding accuracy. (ii) There exists similarities among the libraries' mean algorithms, for instance, the Laplace Truncated mechanism used by diffprivlib is similar to the one used by Google DP (Algorithm 2.4 of Li et al. [56]) in that they both clamp-down outlier outputs. The similarity is reflected in the uniformity of the mean outputs across libraries and, therefore, shows no indication to choose one library over another (excluding Chorus). (iii) The use of diffpriv's sensitivity sampler[3] improves utility substantially. The sensitivity sampler constitutes part of diffpriv's unique selling value proposition, as it

3. For the sensitivity sampler: $\gamma = 0.1$; as per the example provided by the diffpriv team [14]

can calculate the local sensitivity of any query, including machine learning algorithms. However, a query using the sensitivity sampler took roughly 2208 times more to execute than without the sampler for a dataset sizes of 100000 [58].

Sum. Fig. 6 depicts a similar behavior to the mean query. Nonetheless, for higher dataset spreads, Google DP outputs higher and Chorus lower accuracy than in the mean query. This difference in accuracy among libraries disappears as the datasets' spread become smaller.

Var. According to Fig. 7, (i) diffprivlib performs significantly worse in terms of accuracy; however, it is the only library whose outputs are consistent with the var, i.e., its outputs are > 0. diffprivlib achieves output consistency with the Laplace Bounded Domain for the var query at the expense of more noise, i.e., bounding the output domain consumes privacy budget. One may detect diffprivlib's under-performance in the results of MRE, which is lower than Google DP's and SmartNoise's, even when these libraries double the noise executing the mean query twice to calculate the var ($Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$). (ii) Furthermore, the difference in SASE (and in MRE) between the two seemingly matching algorithms of SmartNoise and Google DP comes from the different low-level implementations of the Snapping Laplace mechanism and the use of the Pure Geometric mechanism by SmartNoise. The results show that the combination of their algorithm choice for the var query makes Google DP more accurate for datasets of large spreads and for $\varepsilon \leq 1$, while with lower spreads, SmartNoise provides more utility. (iii) Additionally, Google DP and SmartNoise at least ensure var outputs not to be lower than 0; however, their use of clamping algorithms makes 0 too frequent. (iv) Moreover, SmartNoise and Google DP perform slightly worse regarding precision than the rest of the libraries.

Across all queries. The libraries show similar performance in utility. However, while this is mostly consistent regarding precision, there are notable differences in accuracy in scenarios with datasets of 10000 data points or less and with $\varepsilon \leq 1$ (range recommended by the inventors of DP Dwork et al. [8]). Table 4 shows our library recommendations per query based on their utility performance and other factors such as output consistency. Lastly, except for diffprivlib's count query in the presence of zero-value input data, the MRE results (accuracy) do not show the presence of algorithm bias across libraries.

RQ3. Chorus outperforms the other libraries in memory consumption and diffpriv in execution time, but all libraries are only ready for small-scale deployment.

10. Guidance for Library Designers

We extracted the following actionable advice for library designers, who may also use Tables 2, 3, 4, and 5 for guidance.

Differences in utility. No single library excels at every task (see Table 4), suggesting that library designers can learn from one another. For settings with datasets of 10000 data points or less and with $\varepsilon \leq 1$, we recommend: (i) diffprivlibs Geometric Truncated mechanism for counts (as long as the practitioner does not expect zero values in the dataset). (ii) Google DPs Snapping Laplace mechanism for sums, as it fares better with datasets of a large dataset spread. (iii) Avoid Chorus for the mean, and (iv) employ diffprivlib for the var as it avoids values ≤ 0.

Maturity. As differential privacy becomes more popular, available tools are beginning to demonstrate increasing maturity. diffprivlib, Google DP, and SmartNoise are more advanced tools than Chorus and diffpriv regarding functionality and onboarding practitioners with a basic DP understanding. Moreover, the teams behind diffprivlib (IBM), SmartNoise (Microsoft), and Google DP are responsive to reported bugs and answer questions swiftly in our experience.

Implementation bugs. We call for caution in using any implementation of differential privacy, as existing libraries are relatively young tools. There might still be bugs, such as the one we found in SmartNoise when it was still called WhiteNoise (see Fig. 12), and the one we fixed in Chorus [59]. Additionally, using diffprivs sensitivity sampler on a count query yields an error [60].

Floating point and side channels. diffprivlib, diffpriv, and Chorus and other future library designers should consider implementing the Snapping mechanism. Library designers should also consider addressing other side channels such as privacy accounting and private sensitivity calculation.

Practical value of the Geometric mechanism. Geometric mechanisms are not often used in the differential privacy literature; however, for integer-valued queries, they show considerably better utility (when truncated) than the rest of mechanisms (notably in the count), and it does not have a floating-point vulnerability because its domain is the set of integer numbers.

Output consistency. Except for output rounding in the count query, diffprivlib is the only library that introduces mechanisms such as Laplace Truncated or Bounded Domain for the sum, mean, and var that prevent output inconsistency (e.g., var ≤ 0). All libraries would benefit from the development of improved mechanisms concerning output consistency similar to diffprivlib's.

Operational performance. All libraries offer sufficient performance for small-scale analysis. On the other hand, none of the available libraries appears to be ready for immediate large-scale deployment.

11. Guidance for Researchers

Understanding and communicating theoretical privacy bounds for mechanisms. A practitioner might see that most libraries use a variation of the Laplace mechanism and, consequently, may expect the libraries to agree on how much noise to add; however, our experiments conclude otherwise in specific scenarios (see experiments with $\epsilon < 1$). This is because a privacy guarantee is an *upper* bound on ϵ, not always an exact bound, e.g., a library that adds enough noise for $\varepsilon = 1$ also satisfies DP for $\varepsilon = 2$; practitioners should have in mind that the algorithms' input is an upper bound on ε. In our benchmark, some implementations have proven better than others at achieving *tighter* upper bounds for ε, resulting in more utility. Our results suggest small but important

differences between libraries in terms of the utility obtained for a given ε. Furthermore, features like floating-point protection may yield lower utility for the same input ε, leading to additional confusion for practitioners. Likewise, ensuring var values > 0 may generate more noise than others for the same input ε, yet the added utility is not tangible in the error curves. While these mechanism variations are detailed in their respective literature, the effects on utility may be surprising for a practitioner who is only familiar with the basics of DP. We suggest that additional research is needed on the *concrete* application of DP mechanisms in practice, to understand the gap between theoretical bounds and actual utility and develop ways to communicate that information to practitioners.

Understanding the utility gain and implications of post-processing. The mechanisms benchmarked here are simple, but deployments of differential privacy increasingly make use of more complex mechanisms [31]. These mechanisms increase utility, but this gain is sometimes hard to bound analytically. Moreover, these mechanisms may introduce bias or other artifacts in unpredictable ways, as demonstrated by the results in the 2020 US Census [42]. As more advanced mechanisms make their way to deployments, it is important to understand both the utility gain they provide *and* their other implications on analysis results.

Understanding how much utility to expect. Based on our benchmark, depending on the dataset and the query, the utility may vary, sometimes significantly. More specifically, this dependency is observed for datasets containing at least 10000 data points, and for values of ε lower than 1; Figures 4 and 8 are examples of these conditions, among others in the Appendix. Clearly, utility depends not just on ε, but also on the *scale of the data*. This situation stands in contrast to the way we usually communicate about differential privacy—we typically focus heavily on the setting of ε. This focus is appropriate when communicating with data subjects, whose privacy is of primary concern. However, for analysts attempting to learn from the data, additional guidance is needed about *how to obtain better utility*. In many cases, the answer may be simple—collect more data—but other strategies (like considering alternative mechanisms) may be helpful too. We suggest that additional research is needed to understand how to maximize utility in practice, and to develop effective communication strategies to inform practitioners.

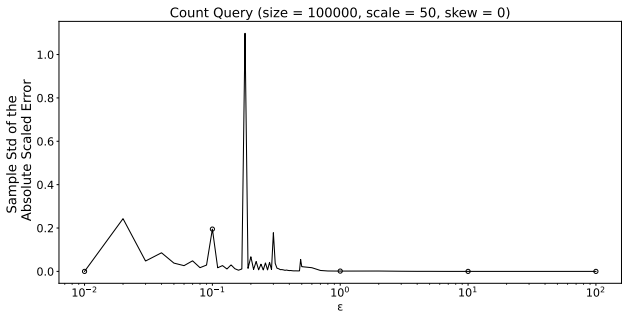

Figure 12: Anomaly caused by a bug encountered in WhiteNoise version 0.1.3 in 2020.

12. Conclusion

To help guide practitioners, library designers, and researchers in navigating the adoption of tools for DP, we performed a qualitative comparison and a benchmark to validate the utility and the scalability of the five libraries towards which we believe a large body of practitioners will gravitate. Based on our results, we recommend practitioners facing a choice among libraries to prioritize protection against side-channels and analyst support; however, there exist enough similarities to feel comfortable choosing any. No single library excels in all aspects, indicating that library designers can learn from one another. In particular, we suggest increasing efforts in output consistency and towards large-scale deployment. We conclude that, as long as practitioners account for potential side-channels, these libraries provide the privacy *they need*.

Limitations. Our benchmark is a snapshot, and our conclusions may become out of date in the future—in fact, we hope they soon will be, due to continued rapid development of the tools! We hope that our findings help library designers (with whom we were in contact throughout our study) improve their products and drive tool support for differential privacy. We also hope that this work highlights potential pitfalls for future library designers. Finally, by releasing the code for our benchmark, we hope this work will form the basis for future evaluations.

Future work. Researchers should consider studying the gap between theoretical bounds and actual utility, researching the practical implications of increasing utility with more complex mechanisms, and discovering other ways to attain more utility. Regarding our benchmark, in particular, researchers may compare SmartNoise and Google DP in their native libraries (Rust and C++, respectively) with our studies' datasets. Moreover, practitioners may benchmark the DP machine learning algorithms proposed by diffprivlib and diffpriv, and others offered by PyTorch and TensorFlow. Furthermore, benchmarking other queries like quantile and mechanisms outside the default set, like the Gaussian and the exponential mechanism, may reveal other obscure differences among libraries. Additionally, other libraries and platforms exist, which, while they did not comply with our inclusion criteria or were recently released, propose DP functionalities that are worth clustering and benchmarking. We have found the following tools: Google's Rappor [61], DJoin [62], ARX [63], PSI [64], Arivat [65], a DP violation detector [66], and Google's ZetaSQL [67]. We do not recommend benchmarking PINQ [16] or GUPT [17] as they are deprecated. Finally, we encourage researchers to select one of the five benchmarked libraries (or a combination) based on this publication and test them in real-world use cases.

References

[1] IBM Security and P. Institue LLC, "2020 cost of a data breach study," p. 82, 2020. [Online]. Available: https://www.ibm.com/security/data-breach

[2] L. Sweeney, A. Abu, and J. Winn, "Identifying Participants in the Personal Genome Project by

Name," *SSRN Electronic Journal*, 2013. [Online]. Available: http://www.ssrn.com/abstract=2257732

[3] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang, and J. Lindqvist, "Elastic pathing: your speed is enough to track you," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. Seattle, Washington: ACM Press, 2014, pp. 975–986. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2632048.2632077

[4] D. Kondor, B. Hashemian, Y.-A. de Montjoye, and C. Ratti, "Towards Matching User Mobility Traces in Large-Scale Datasets," *IEEE Transactions on Big Data*, vol. 6, no. 4, pp. 714–726, Dec. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8470173/

[5] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. Oakland, CA, USA: IEEE, May 2008, pp. 111–125, iSSN: 1081-6011. [Online]. Available: http://ieeexplore.ieee.org/document/4531148/

[6] M. Archie, S. Gershon, A. Katcoff, and A. Zeng, "De-anonymization of Netix Reviews using Amazon Reviews," p. 5, 2018.

[7] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: Threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[8] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! A Survey of Attacks on Private Data," *Annual Review of Statistics and Its Application*, vol. 4, no. 1, pp. 61–84, Mar. 2017. [Online]. Available: http://www.annualreviews.org/doi/10.1146/annurev-statistics-060116-054123

[9] Microsoft, "SmartNoise repository," https://github.com/opendp/smartnoise-core, online; accessed 21 September 2021.

[10] IBM, "diffprivlib repository," https://github.com/IBM/differential-privacy-library, online; accessed 21 September 2021.

[11] Google, "Google DP repository," https://github.com/google/differential-privacy, online; accessed 21 September 2021.

[12] Joseph P. Near, "Chorus repository," https://github.com/uvm-plaid/chorus, online; accessed 21 September 2021.

[13] N. Johnson, J. P. Near, J. M. Hellerstein, and D. Song, "Chorus: Differential Privacy via Query Rewriting," *arXiv:1809.07750 [cs]*, Sep. 2018, arXiv: 1809.07750. [Online]. Available: http://arxiv.org/abs/1809.07750

[14] Rubinstein, Benjamin, "diffpriv repository," https://github.com/brubinstein/diffpriv, online; accessed 21 September 2021.

[15] B. I. P. Rubinstein and F. Ald, "Pain-Free Random Differential Privacy with Sensitivity Sampling," *arXiv:1706.02562 [cs, stat]*, Jun. 2017, arXiv: 1706.02562. [Online]. Available: http://arxiv.org/abs/1706.02562

[16] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1930. [Online]. Available: https://doi.org/10.1145/1559845.1559850

[17] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "GUPT: privacy preserving data analysis made easy," in *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12*. Scottsdale, Arizona, USA: ACM Press, 2012, p. 349. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2213836.2213876

[18] Munilla, Gonzalo and Muhammad, Aitsam, "Benchmark repository," https://github.com/gonzalo-munillag/Benchmarking_Differential_Privacy_Analytics_Libraries, online; accessed 21 September 2021.

[19] I. Mironov, "On significance of the least significant bits for differential privacy," in *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. Raleigh, North Carolina, USA: ACM Press, 2012, p. 650. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2382196.2382264

[20] C. Dwork, "Differential Privacy," 2006, in: Bugliesi M., Preneel B., Sassone V., Wegener I. (eds) Automata, Languages and Programming. Lecture Notes in Computer Science, vol 4052. Springer, Berlin, Heidelberg.

[21] M. Hardt and K. Talwar, "On the geometry of differential privacy," in *Proceedings of the forty-second ACM symposium on Theory of computing*, 2010, pp. 705–714.

[22] J. Lee and C. Clifton, "How much is enough? Choosing ϵ for differential privacy," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7001 LNCS, 2011, pp. 325–340.

[23] A. De, "Lower bounds in differential privacy," in *Theory of cryptography conference*. Springer, 2012, pp. 321–338.

[24] A. Friedman and A. Schuster, "Data mining with differential privacy." Washington, DC, USA: ACM Press, 2010, p. 493, proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10.

[25] Y. Yang, Z. Zhang, G. Miklau, M. Winslett, and X. Xiao, "Differential privacy in data publication and analysis," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 601606. [Online]. Available: https://doi.org/10.1145/2213836.2213910

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[27] X. Xiong, S. Liu, D. Li, Z. Cai, and X. Niu, "A comprehensive survey on local differential privacy," *Security and Communication Networks*, vol. 2020, 2020.

[28] M. Yang, L. Lyu, J. Zhao, T. Zhu, and K.-Y. Lam, "Local differential privacy and its applications: A comprehensive survey," 2020.

[29] M. U. Hassan, M. H. Rehmani, and J. Chen, "Differential privacy techniques for cyber physical systems: a survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 746–789, 2019.

[30] B. Nelson and J. Reuben, "Sok: Chasing accuracy and privacy, and catching both in differentially private histogram publication," 2020.

[31] C. Li, M. Hay, G. Miklau, and Y. Wang, "A data- and workload-aware algorithm for range queries under differential privacy," *Proc. VLDB Endow.*, vol. 7, no. 5, p. 341352, Jan. 2014. [Online]. Available: https://doi-org.eaccess.ub.tum.de/10.14778/2732269.2732271

[32] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 2013, pp. 757–768.

[33] W. Qardaji, W. Yang, and N. Li, "Understanding hierarchical methods for differentially private histograms," *Proc. VLDB Endow.*, vol. 6, no. 14, p. 19541965, Sep. 2013. [Online]. Available: https://doi-org.eaccess.ub.tum.de/10.14778/2556549.2556576

[34] H. Zhang, E. Roth, A. Haeberlen, B. C. Pierce, and A. Roth, "Testing differential privacy with dual interpreters," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–26, Nov. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3428233

[35] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang, "Principled Evaluation of Differentially Private Algorithms using DPBench," in *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*. San Francisco, California, USA: ACM Press, 2016, pp. 139–154. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2882903.2882931

[36] F. T. Wu, "Defining privacy and utility in data sets," *84 University of Colorado Law Review 1117 (2013); 2012 TRPC*, pp. 1117–1177, 2012.

[37] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 486–503.

[38] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042

[39] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 193–204.

[40] C. Dwork, "Differential Privacy: A Survey of Results," in *Theory and Applications of Models of Computation*, M. Agrawal, D. Du, Z. Duan, and A. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4978, pp. 1–19, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-540-79228-4_1

[41] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, 2007, pp. 94–103.

[42] S. L. Garfinkel, J. M. Abowd, and S. Powazek, "Issues encountered deploying differential privacy," in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, ser. WPES'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 133137. [Online]. Available: https://doi-org.eaccess.ub.tum.de/10.1145/3267323.3268949

[43] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[44] Scipy community, "Skew-normal distribution documentation," https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skewnorm.html, online; accessed 21 September 2021.

[45] A. Azzalini and A. Capitanio, "Statistical applications of the multivariate skew normal distribution," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, p. 579602, Aug 1999. [Online]. Available: http://dx.doi.org/10.1111/1467-9868.00194

[46] Kaggle, "Census dataset repository," https://www.kaggle.com/muonneutrino/us-census-demographic-data, online; accessed 21 September 2021.

[47] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[48] UCI machine learning repository, "Education dataset repository," https://archive.ics.uci.edu/ml/datasets/Student+Performance, online; accessed 21 September 2021.

[49] P. Cortez, A. Silva, and D. Andrade, "Predicting academic achievement of high-school students using machine learning." 2014, psychology, 5, 2046-2057.

[50] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally Utility-Maximizing Privacy Mechanisms," *arXiv:0811.2841 [cs]*, Mar. 2009, arXiv: 0811.2841. [Online]. Available: http://arxiv.org/abs/0811.2841

[51] R. Bassily, A. Smith, and A. Thakurta, "Differentially private empirical risk minimization: Efficient algorithms and tight error bounds," 2014.

[52] Q. Geng and P. Viswanath, "The Optimal Mechanism in Differential Privacy," *arXiv:1212.1186 [cs]*, Oct. 2013, arXiv: 1212.1186. [Online]. Available:

http://arxiv.org/abs/1212.1186

[53] N. Holohan, S. Antonatos, S. Braghin, and P. Mac Aonghusa, "The Bounded Laplace Mechanism in Differential Privacy," *Journal of Privacy and Confidentiality*, vol. 10, no. 1, Dec. 2019.

[54] B. Rubinstein and F. Alda, "diffpriv: An R Package for Easy Differential Privacy," p. 5, 2017. [Online]. Available: https://cran.r-project.org/web/packages/diffpriv/vignettes/diffpriv.pdf

[55] Microsoft and Harvard, "SmartNoise' sensitivity analytical proofs," https://github.com/opendifferentialprivacy/SmartNoise-core/tree/develop/whitepapers/sensitivities, online; accessed 21 September 2021.

[56] N. Li, M. Lyu, D. Su, and W. Yang, *Differential Privacy: From Theory to Practice*, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7731575

[57] Intel, "Xeon processor," https://ark.intel.com/content/www/de/de/ark/products/75269/intel-xeon-processor-e5-2650-v2-20m-cache-2-60-ghz.html, online; accessed 21 September 2021.

[58] Munilla, Gonzalo and Muhammad, Aitsam, "diffpriv's sensitivity sampler excessive execution time," https://github.com/gonzalo-munillag/Benchmarking_Differential_Privacy_Analytics_Libraries/tree/main/Time_and_Memory_calculation/Code/R/sensitivity_sampler_execution_time, online; accessed 21 September 2021.

[59] Muhammad, Aitsam, "Chorus bug fix," https://github.com/uvm-plaid/chorus/pull/1/commits/b20da14849bb889647418653afc6090a4f2a9c85, online; accessed 21 September 2021.

[60] Munilla, Gonzalo and Muhammad, Aitsam, "diffpriv's sensitivity sampler count error," https://github.com/gonzalo-munillag/Benchmarking_Differential_Privacy_Analytics_Libraries/tree/main/Bugs/diffpriv, online; accessed 21 September 2021.

[61] . Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Scottsdale Arizona USA: ACM, Nov. 2014, pp. 1054–1067. [Online]. Available: https://dl.acm.org/doi/10.1145/2660267.2660348

[62] A. Narayan and A. Haeberlen, "DJoin: Differentially Private Join Queries over Distributed Databases," p. 14, 2012.

[63] R. Bild, K. A. Kuhn, and F. Prasser, "Safepub: A truthful data anonymization algorithm with strong privacy guarantees," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 67–87, 2018. [Online]. Available: https://doi.org/10.1515/popets-2018-0004

[64] B. Kacsmar, B. Khurram, N. Lukas, A. Norton, M. Shafieinejad, Z. Shang, Y. Baseri, M. Sepehri, S. Oya, and F. Kerschbaum, "Differentially Private Two-Party Set Operations," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. Genoa, Italy: IEEE, Sep. 2020, pp. 390–404. [Online]. Available: https://ieeexplore.ieee.org/document/9230399/

[65] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and Privacy for MapReduce," p. 16, 2010.

[66] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "Detecting Violations of Differential Privacy," p. 15, 2019.

[67] Google, "ZetaSQL repository," https://github.com/google/zetasql, online; accessed 21 September 2021.

[68] Q. Geng, W. Ding, R. Guo, and S. Kumar, "Privacy and Utility Tradeoff in Approximate Differential Privacy," *arXiv:1810.00877 [cs]*, Feb. 2019, arXiv: 1810.00877. [Online]. Available: http://arxiv.org/abs/1810.00877

[69] B. Balle and Y.-X. Wang, "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising," *arXiv:1805.06530 [cs, stat]*, Jun. 2018, arXiv: 1805.06530. [Online]. Available: http://arxiv.org/abs/1805.06530

[70] C. L. Canonne, G. Kamath, and T. Steinke, "The Discrete Gaussian for Differential Privacy," *arXiv:2004.00010 [cs, stat]*, Jan. 2021, arXiv: 2004.00010. [Online]. Available: http://arxiv.org/abs/2004.00010

[71] R. McKenna and D. Sheldon, "Permute-and-Flip: A new mechanism for differentially private selection," *arXiv:2010.12603 [cs]*, Oct. 2020, arXiv: 2010.12603. [Online]. Available: http://arxiv.org/abs/2010.12603

[72] C. Ilvento, "Implementing the Exponential Mechanism with Base-2 Differential Privacy," *arXiv:1912.04222 [cs]*, Aug. 2020, arXiv: 1912.04222. [Online]. Available: http://arxiv.org/abs/1912.04222

[73] N. Holohan, D. J. Leith, and O. Mason, "Optimal Differentially Private Mechanisms for Randomised Response," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2726–2735, Nov. 2017, arXiv: 1612.05568. [Online]. Available: http://arxiv.org/abs/1612.05568

[74] J. T. Kent, A. M. Ganeiber, and K. V. Mardia, "A New Unified Approach for the Simulation of a Wide Class of Directional Distributions," *Journal of Computational and Graphical Statistics*, vol. 27, no. 2, pp. 291–301, Apr. 2018. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10618600.2017.1390468

Appendix

1. Experiments workflow

Algorithm 1 Logic of the experiments to measure utility from the outputs of the analytics queries.

Input: $\mathcal{P}$, set of libraries to benchmark; $\mathcal{M}_{\mathcal{P}}$, set of randomized analytics queries to benchmark within a library; $\mathcal{D}$, set of datasets; $\mathcal{E}$, set of ε; $\mathcal{N}$, number of experiments for a single ε; $\mathcal{W}_{\mathcal{M}_{\mathcal{P}}}$, set of baseline truthful analytics queries mapped to a given $\mathcal{M}_{\mathcal{P}}$.

Output: $\hat{y}$, noisy query result; y, truthful query result; E, set of size $\mathcal{N}$ of L_1 errors; RE, set of size $\mathcal{N}$ of L_1 relative errors; $\widehat{RE}$, set of cardinality $|\mathcal{E}|$ of sample means of each RE; S, set of cardinality $|\mathcal{E}|$ of sample std of each scaled E.

for each: $p \in \mathcal{P}$
  for each: $\mathcal{M}_p \in \mathcal{M}_{\mathcal{P}}$
    for each: $\mathcal{I} \in \mathcal{D}$
      $\widehat{RE}$ = {}; S = {};
      for each: $\varepsilon \in \mathcal{E}$
        RE = {}; E = {}
        $y = \mathcal{W}_{\mathcal{M}_p}(\mathcal{I}, \varepsilon)$
        for i = 1 to $\mathcal{N}$ **do**
          $\hat{y} = \mathcal{M}_p(\mathcal{I}, \varepsilon)$
          $E = E \cup \|\hat{y} - y\|_1$
          $RE = RE \cup \frac{\|\hat{y} - y\|_1}{\|y\|_1}$
        $\widehat{RE} = \widehat{RE} \cup mean(RE)$
        $S = S \cup SampleStd\left(\frac{E}{|\mathcal{I}|}\right)$
      plot($\widehat{RE}$, $\mathcal{E}$)
      plot(S, $\mathcal{E}$)

2. Differential privacy mechanisms

3. Experiment Results

TABLE 5: Overview of all the mechanisms implemented by the benchmarked libraries, among others.

Mechanism	Implementation	Libraries	Description
Laplace	Pure	All	Noise is sampled from a Laplace distribution of domain $(-\inf, \inf)$. This noise is added to the true value without post-processing. [38]
	Truncated	diffprivlib	If after adding noise to the true value this output falls outside a pre-defined range, then the output is mapped to the closest bound of the output range, e.g. a count output of less than zero could be mapped to the lower bound 0. If the domain bounds coincide with e.g. changes in behavior because the probability of returning values at the domain bounds is non-zero, it is recommended to use the Bounded Domain mechanism instead [53].
	Bounded Domain	diffprivlib	Samples outputs until one falls within the pre-defined range. While this mechanism is suitable to prevent not desired values, e.g. a negative var value, or implemented in classifiers, it requires more tailoring so that it satisfies DP [53].
	Bounded Noise	diffprivlib	Samples from a truncated domain of a Laplacian distribution [68].
	Folded	diffprivlib	Similar to Truncated, but instead of outputting the closest bound, the output outside a pre-defined range is folded around this range until the output falls within, e.g. with a pre-described range of $[l, u]$, a count output of $c < l$ could be folded by recursively computing $n * l - (c)$ until the count falls within $[l, u]$, where $n \in \mathbb{N}$.
Gaussian	Pure	All including Google-DP, but except PyDP	Noise is sampled from a Gaussian distribution of domain $(-\inf, \inf)$. This noise is added to the true value without post-processing. [38]
	Analytic Gaussian	diffprivlib	It removes at least a third of the variance of the Pure Gaussian mechanism by a novel noise calibration strategy and a post-processing technique [69].
	Discrete Gaussian	diffprivlib	Modifies the Gaussian mechanism so that discrete noise may be sampled without losing privacy or accuracy guarantees [70].
Exponential	Pure	All including Google-DP, but except PyDP	Achieves differential privacy for categorical query outputs by randomly choosing a category proportionally to its utility value [41], i.e. a category with a higher utility score than another is as much more likely to be picked. the utility values decay exponentially, making the true results more likely to be picked while maintaining DP.
	Permute and Flip	diffprivlib	It randomly selects a value, then, the algorithm flips a biased coin based on the utility of the selected value [71], and releases the output if the results it heads.
	Hierarchical	diffprivlib	Adapts the Pure exponential mechanism to hierarchical data, so that the utility calculation is less complex due to the intrinsic hierarchy of the data.
	With Base-2 DP	SmartNoise (In progress)	By switching the DP definition from base e to base 2, one is able to perform precise base 2 arithmetic, and, thus, avoid floating-point vulnerability. [72].
Geometric	Pure	diffprivlib, SmartNoise	Employs a discrete variant of the Laplace mechanism by satisfying DP with equality, and thus, producing tighter guarantees for integer-value outputs, and, in turn, higher accuracy [50].
	Truncated	diffprivlib	Uses the same technique as Laplace Truncated but the underlying mechanism is the Geometric.
	Folded	diffprivlib	Uses the same technique as Laplace Folded but the underlying mechanism is the Geometric.
Staircase	Pure	diffprivlib	Optimizes the Laplace Mechanism, obtaining more accuracy for moderate-low ε values. The shape of the noise distribution is a *staircase*, technically considered as a *geometric mixture of uniform probability distributions.* [52]
Binary	Pure	diffprivlib	Specifically designed for binary inputs, this mechanism adapts randomized response to (ε, δ)-differential privacy. In effect, the logic flips a biased coin to output the true input or its complementary [73].
Bingham	Pure	diffprivlib	A differentially-private mechanism built upon the Bingham distribution, exclusively used for estimating the first eigenvector of a covariance matrix. [74]
Vector	Pure	diffprivlib	Employed for perturbing convex objective functions of a machine learning classifier before optimization [51].
Uniform	Pure	diffprivlib	Derived from the edge case where the Laplace Bounded Noise has an ε = 0 [68].
Snapping	Laplace	SmartNoise, Google-DP	A modification of the Laplace mechanism to protect against the floating-point vulnerability of the theoretical Laplace continuous distribution. Among other steps, the key of the mechanism is rounding to the closest multiple of a power of 2. Furthermore, note that the Snapping mechanism implementation from SmartNoise and Google-DP differ, and in turn, both differ from the original work from [19], as they achieve floating-point safety while adding less noise.
	Gaussian	Google-DP	Similar to the Snapping mechanism for the Laplace distribution, this variation modifies the Gaussian mechanism to protect against the floating-point vulnerability of the theoretical Gaussian continuous distribution.[19]

TABLE 6: Set of detailed experiments' outputs for all three independent variables (dataset size, scale, and skewness) of the sum query (500 experiments per ε).

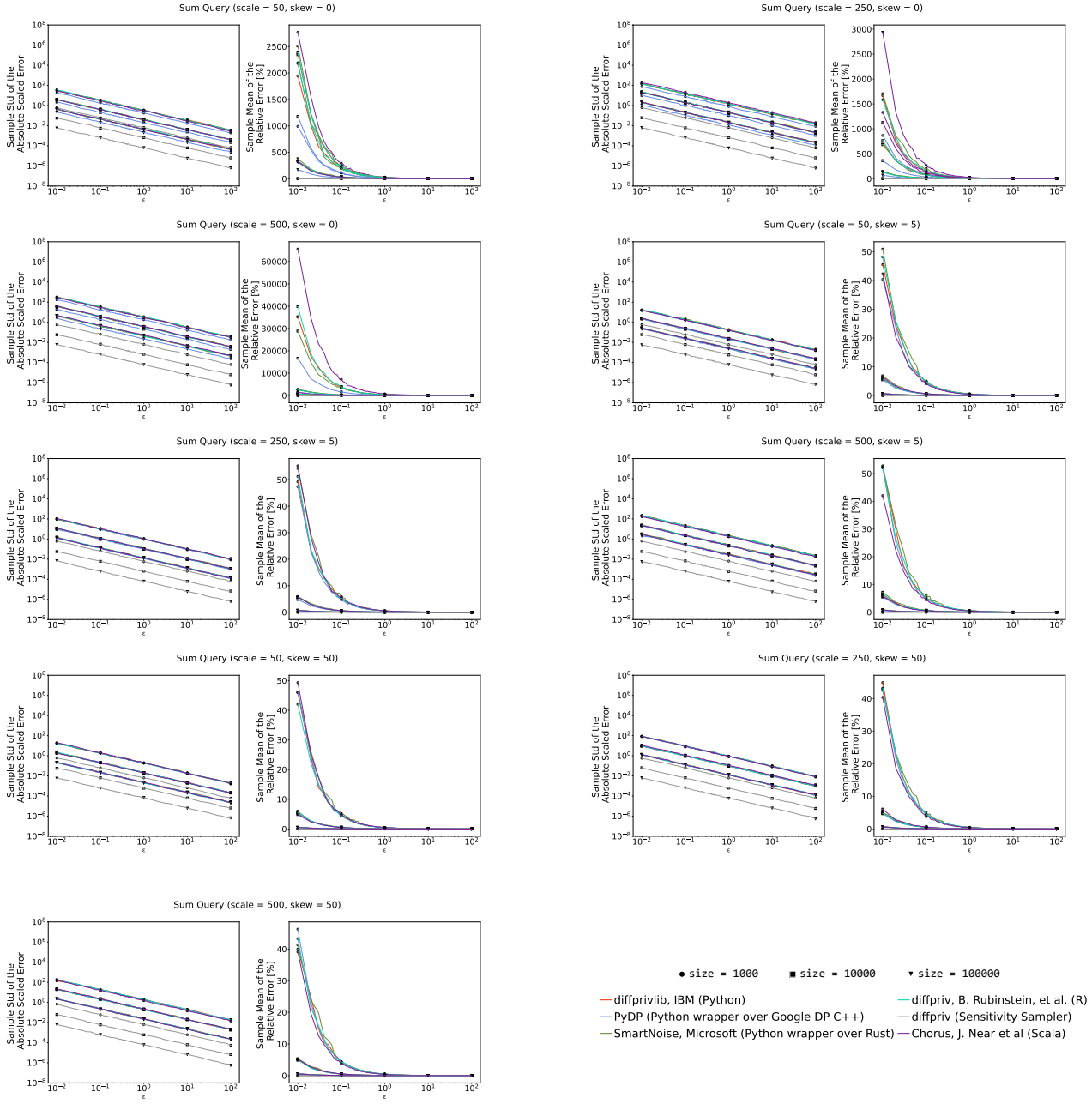

TABLE 7: Set of detailed experiments' outputs for all three independent variables (dataset size, scale, and skewness) of the mean query (500 experiments per ε).

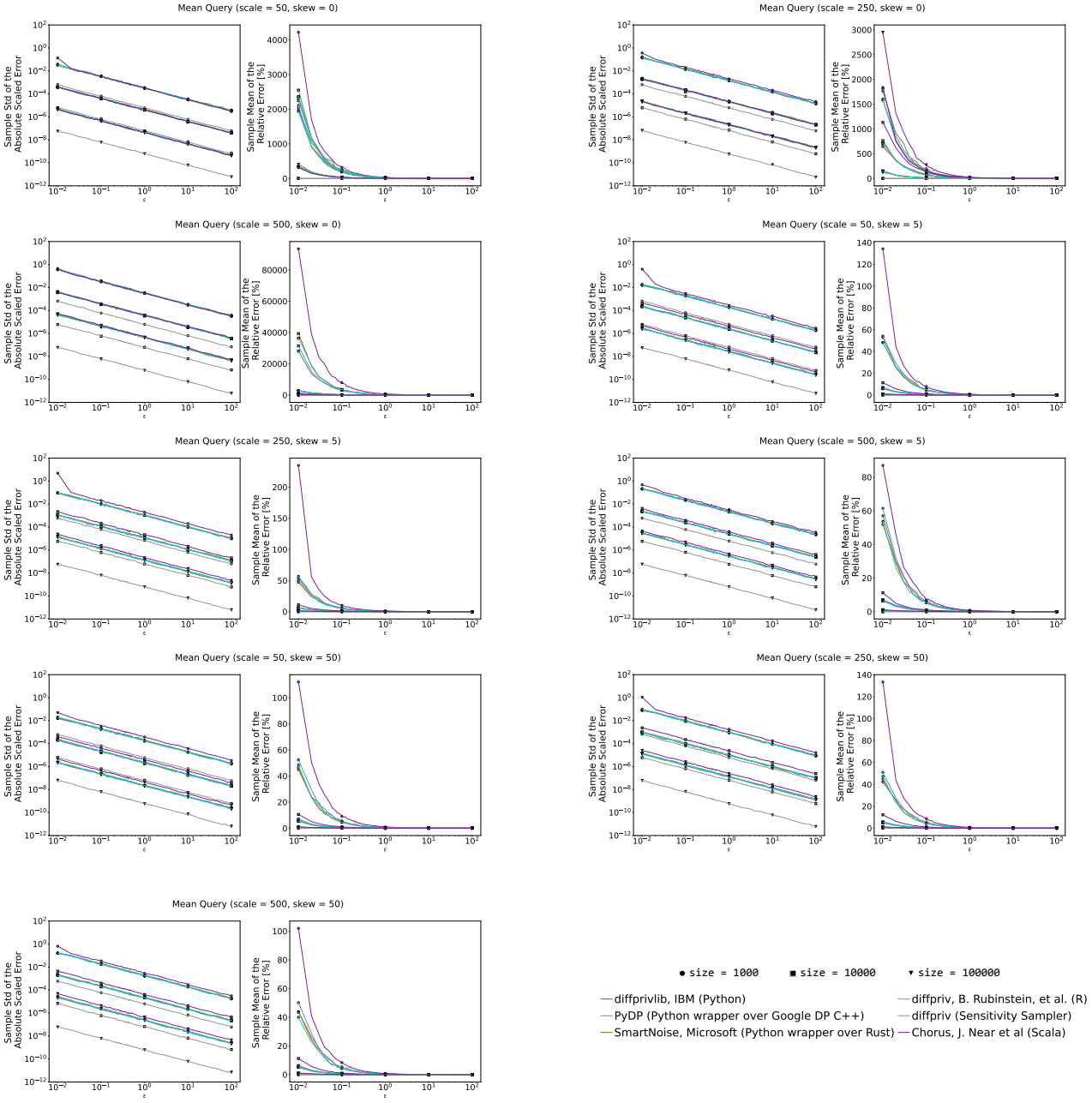

TABLE 8: Set of detailed experiments' outputs for all three independent variables (dataset size, scale, and skewness) of the var query (500 experiments per ε).

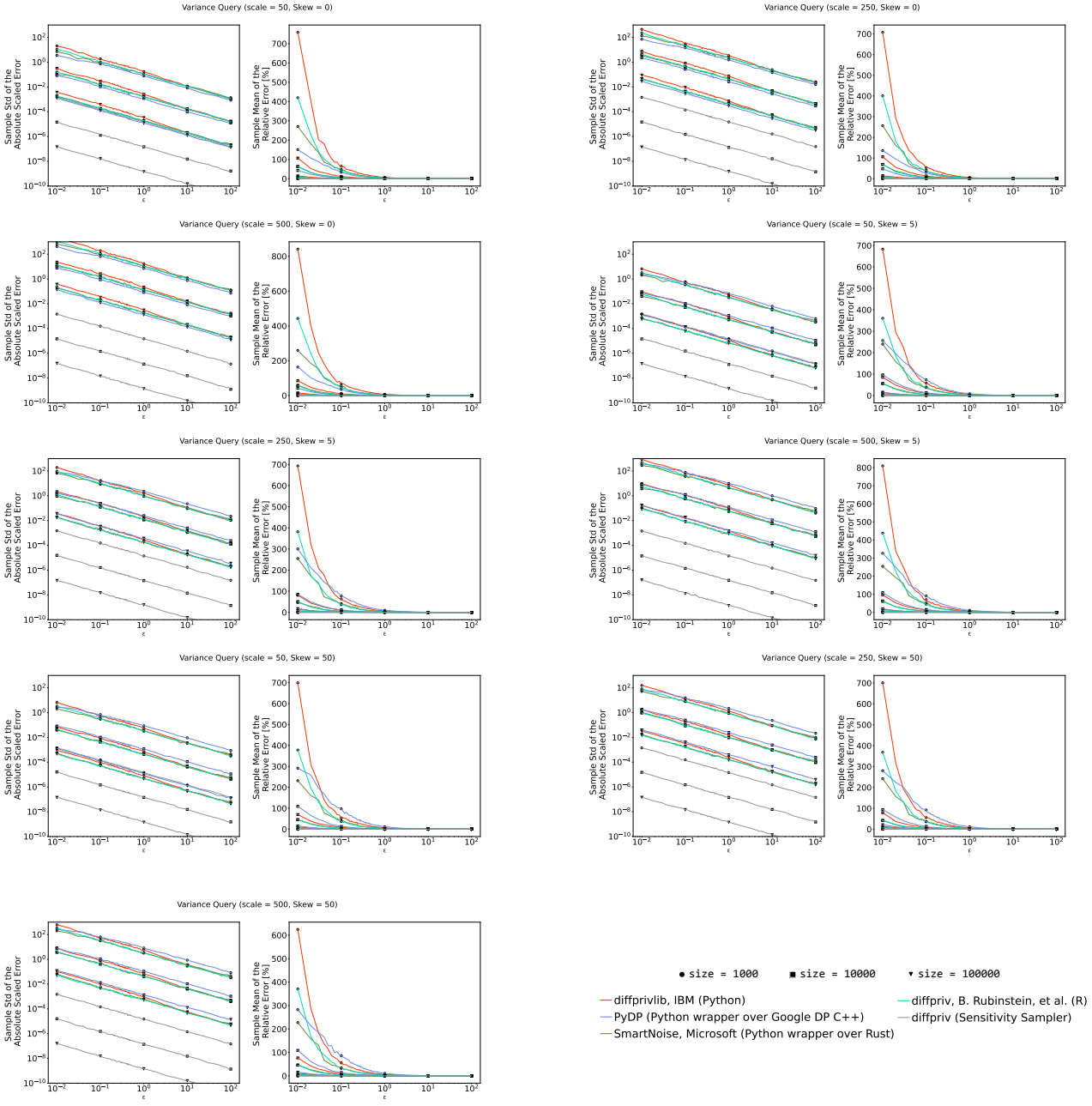

TABLE 9: Sample std of the absolute scaled error across varying dataset scales (5 experiments per ε and dataset scale).

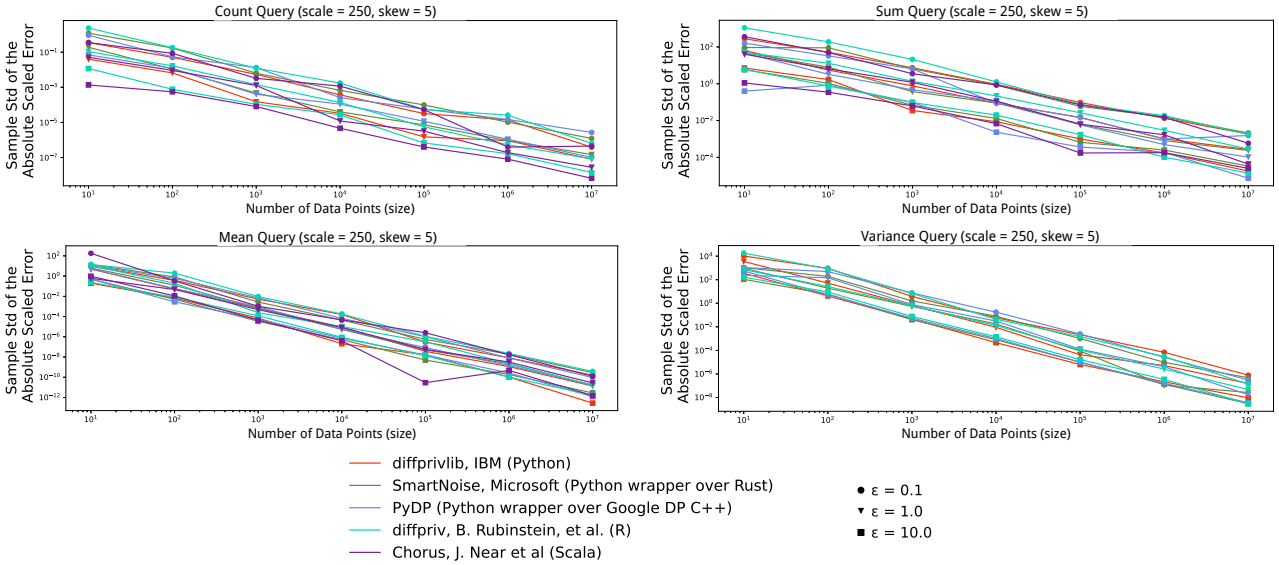

TABLE 10: Execution time for a range of an increasing number of data points for all queries: count, sum, mean, and var (5 experiments per ε and dataset size).

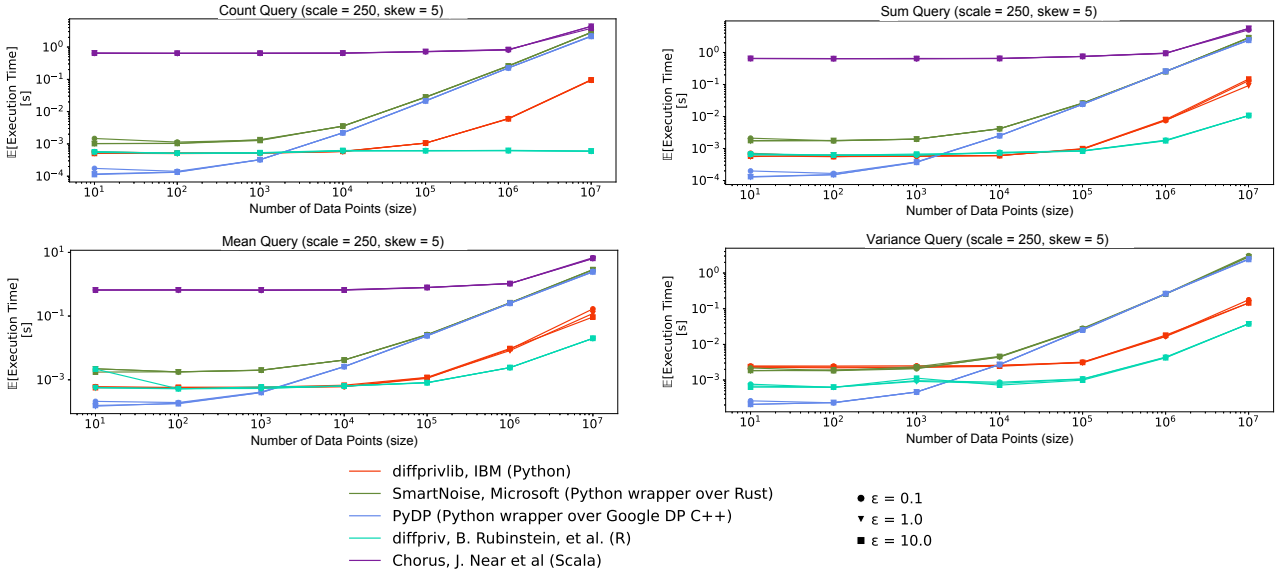

TABLE 11: Memory consumption for a range of an increasing number of data points for all queries: count, sum, mean, and var (5 experiments per ε and dataset size).

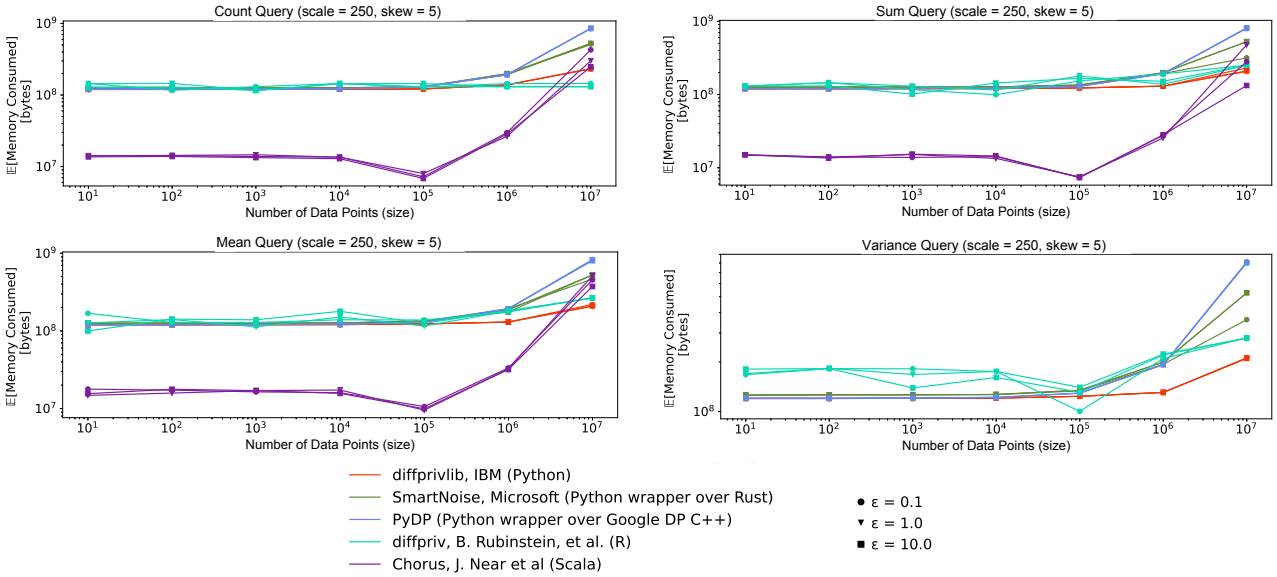

TABLE 12: Experiments of the queries count, sum, mean, and var on the attribute Age of the U.S.A census dataset containing 48842 individuals (500 experiments per ε).

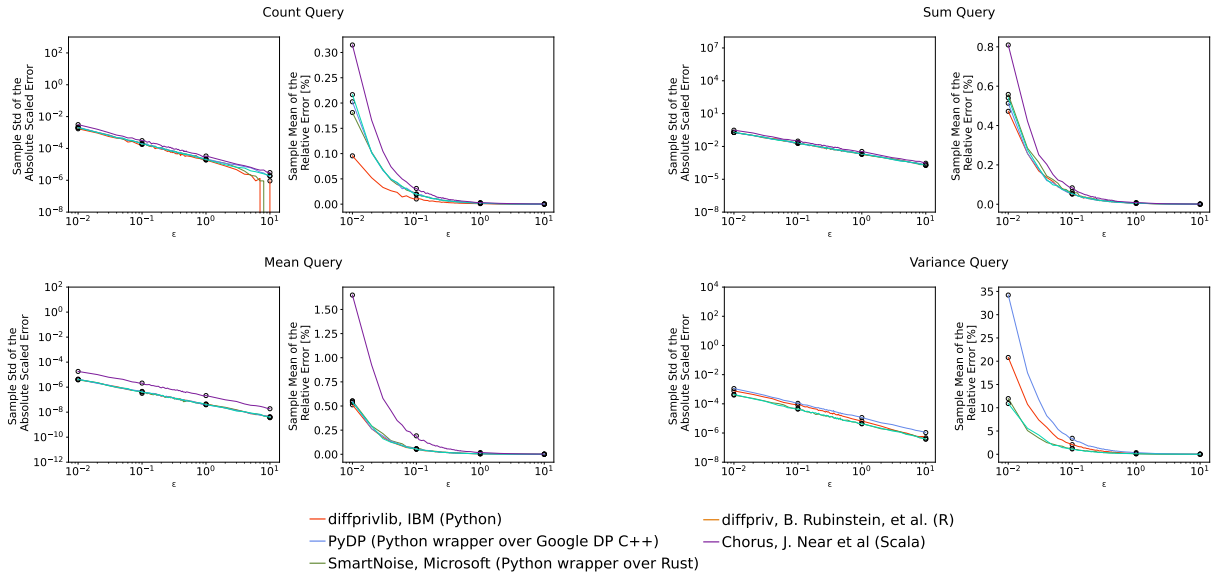

TABLE 13: Experiments of the queries count, sum, mean, and var on the attribute Hours of the U.S.A census dataset containing 48842 individuals (500 experiments per ε).

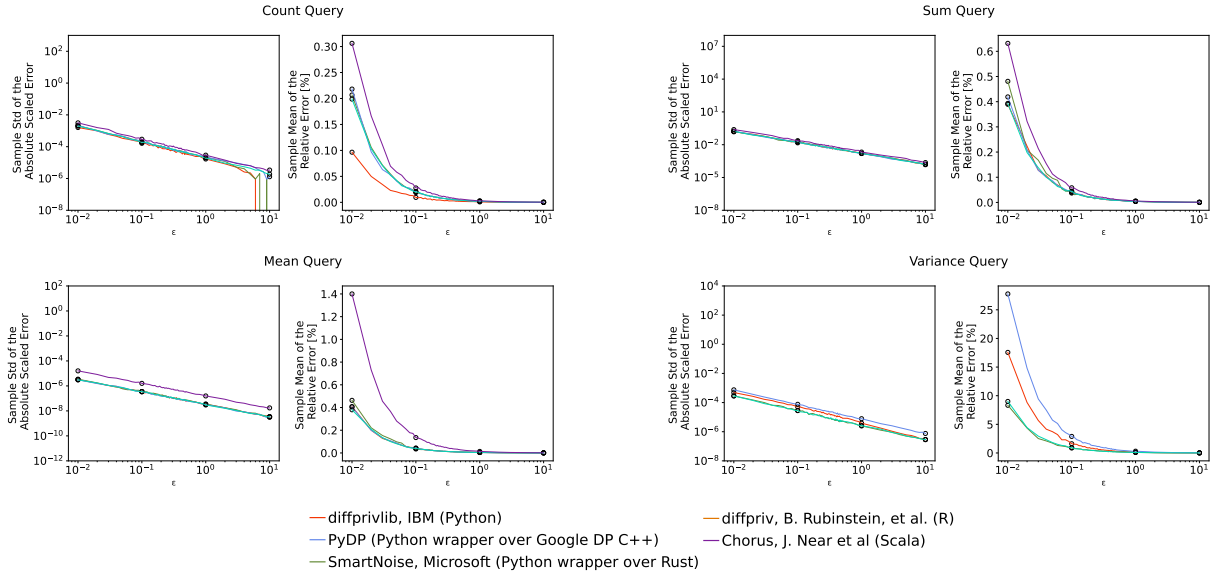

TABLE 14: Experiments of the queries count, sum, mean, and var on the attribute Absences of the Portuguese education dataset containing 649 individuals (500 experiments per ε).

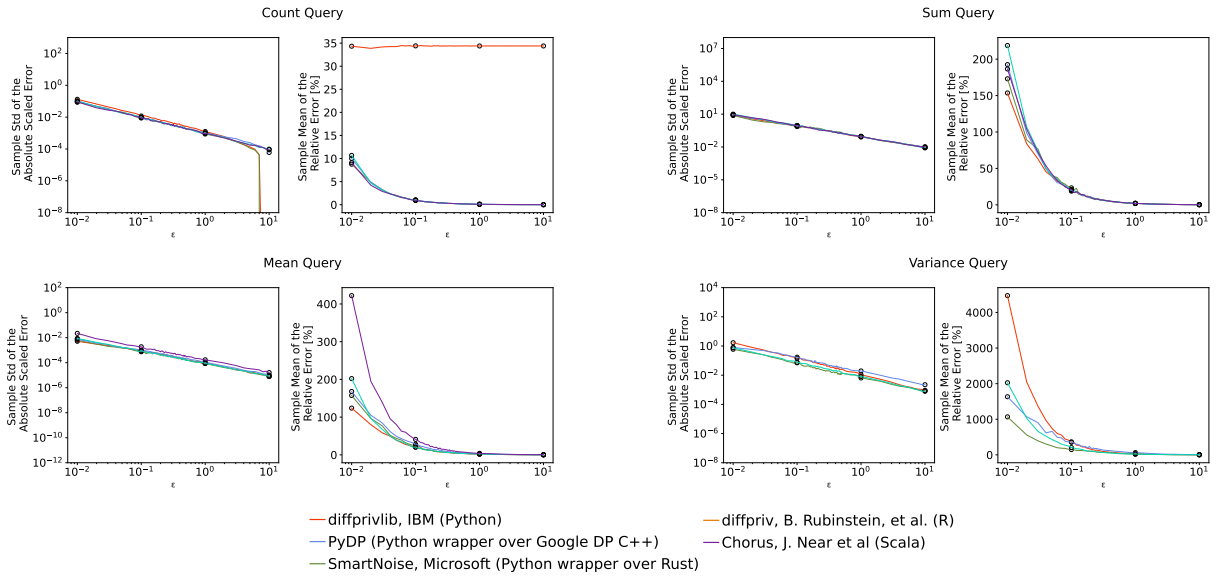

TABLE 15: Experiments of the queries count, sum, mean, and var on the attribute Grades of the Portuguese education dataset containing 649 individuals (500 experiments per ε).