Pushing the Boundaries of Private, Large-Scale Query Answering

Brendan Avent[*] and Aleksandra Korolova[**]

[*]University of Southern California
bavent@usc.edu
[**]Princeton University
korolova@princeton.edu

Abstract

We address the problem of efficiently and effectively answering large numbers of queries on a sensitive dataset while ensuring differential privacy (DP). We separately analyze this problem in two distinct settings, grounding our work in a state-of-the-art DP mechanism for large-scale query answering: the Relaxed Adaptive Projection (RAP) mechanism.

The first setting is a classic setting in DP literature where all queries are known to the mechanism in advance. Within this setting, we identify challenges in the RAP mechanism's original analysis, then overcome them with an enhanced implementation and analysis. We then extend the capabilities of the RAP mechanism to be able to answer a more general and powerful class of queries (r-of-k thresholds) than previously considered. Empirically evaluating this class, we find that the mechanism is able to answer orders of magnitude larger sets of queries than prior works, and does so quickly and with high utility.

We then define a second setting motivated by real-world considerations and whose definition is inspired by work in the field of machine learning. In this new setting, a mechanism is only given partial knowledge of queries that will be posed in the future, and it is expected to answer these future-posed queries with high utility. We formally define this setting and how to measure a mechanism's utility within it. We then comprehensively empirically evaluate the RAP mechanism's utility within this new setting. From this evaluation, we find that even with weak partial knowledge of the future queries that will be posed, the mechanism is able to efficiently and effectively answer arbitrary queries posed in the future. Taken together, the results from these two settings advance the state of the art on differentially private large-scale query answering.

Contents

1 Overview

Many data analysis and machine learning algorithms, at their core, involve answering *statistical queries*. Statistical queries are the class of queries that answer the question: "What fraction of entries in a given dataset have a particular property P?" Because of their ubiquity, developing differentially private mechanisms to effectively answer statistical queries has been one of the most well studied problems in DP [DN03, BDMN05, DMNS06, BLR08, DNR+09, DRV10, RR10, HR10, HLM12, GRU12]. Early DP research primarily focused on designing mechanisms to answer specific, individual statistical queries in an interactive setting. In that setting, queries are posed and answered one at a time with the goal of answering each query with minimal error while ensuring privacy. However, most practical data-driven algorithms do not pose only a single query. Instead, they pose a large number of queries, referred to as a *query workload*. When a query workload is available in advance (i.e., prespecified), it is possible to design DP mechanisms that take advantage of the relationships between the queries to achieve higher utility relative to answering the individual queries independently. In this work, we address the problem of privately answering a large number of queries by answering the following high-level research question.

In the two following settings, to what extent are differentially private mechanisms able to answer a large number of statistical queries efficiently and with low error?

Setting 1: All queries are prespecified; i.e., known in advance.

Setting 2: Only partial knowledge of the queries is available in advance.

Motivating Example

A motivating data analysis example for this work is The American Community Survey (ACS), a demographics survey program conducted by the U.S. Census Bureau [Bur16]. The ACS regularly gathers information such as ancestry, citizenship, educational attainment, income, language proficiency, migration, disability, employment, and housing characteristics. The Census Bureau aggregates the individual ACS responses (microdata), then generates population estimates which are available to the public via online data tools. The most popular tool, Public Use Microdata Sample (PUMS), enables researchers to generate custom cross-tabulations of responses to the ACS questions. To protect the privacy of the ACS respondents, PUMS data are sampled, anonymized, and only available for sufficiently populous geographics regions. However, studies have found that the ad hoc anonymization techniques used are not entirely sufficient to protect the privacy of individual respondents (e.g., via re-identification attacks) [Abo18, CRB22]. As a result, the Census Bureau has announced plans to incorporate differential privacy into the American Community Survey, and declared that it is researching "a new fully-synthetic data product" with a development period ending in 2025 [Rod21, Dai22].

One promising and active direction within DP research is synthetic data generation [MSM19, VTB+20, LVW21]. The hope is that once a synthetic dataset is generated via a differentially private mechanism, researchers and analysts can pose an arbitrary number of queries against the synthetic dataset without increasing the privacy risk to those who contributed the original underlying data. DP synthetic data generation mechanisms seek to strike a balance between distilling the information in the underlying dataset most useful to analysts while simultaneously ensuring privacy of the underlying dataset. Thus, to maximize the eventual usefulness of the synthetic dataset, synthetic data generation mechanisms must tailor the generated dataset to the specific class of downstream tasks (e.g., a particular class of queries) that analysts are most likely interested in. This is typically done by providing a set of queries (the query workload) to the DP mechanism, so that the mechanism can tailor the synthetic dataset to answering these queries (and, ideally, to other similar queries). Much of DP synthetic data research has focused on designing mechanisms to generate synthetic data which can provide accurate answers (under a variety of metrics, most commonly ℓ_∞ error) to the subset of statistical queries known as k-way marginal queries [BCD+07, TUV12, GHRU13, CTUW14, CKS18, MSM19, VTB+20, NBRS22]. Informally, a k-way marginal query is one which answers the question: "What fraction of people in the private dataset have all of the following k attributes: ...?" In this work, we focus on a strict generalization of k-way marginal queries known as r-of-k threshold queries [KLPV87, Lit88, HW04, TUV12, Ull13, ABK+21] under the ℓ_∞ error metric. Informally, r-of-k threshold queries answer the question: "What fraction of people in the private dataset have at least r of the following k attributes: ...?".

As a simplified example of where such queries can be used, we consider the scenario where a social scientist is interested in using ACS data to determine what portion of a community has a substandard quality of living. Suppose the scientist wants to examine the four following attributes for each person in the community: is their income level below the poverty line, are they unemployed, are they homeless, do they have a low net worth? Clearly, a person having any single attribute does not necessarily mean that they have a substandard quality of living. Similarly, a person does not need to have all four attributes to have a substandard quality of living. Thus, the social scientist can formulate this as an r-of-k threshold query with $r = 3$, $k = 4$; i.e., a person has a substandard quality of living if they have at least three of the four attributes.

This social scientist may have many such queries, and other researchers may have sets of queries of their own that they wish to pose. Thus, a natural algorithm design question is: how should the U.S. Census Bureau answer everyone's queries with low error while still ensuring the ACS respondents' privacy? The simplest option is to use a portion of the DP budget to individually answer each query, independent of all other queries. This would likely be unsatisfactory utility-wise, since it both limits how many queries can be answered and ignores any relationships between queries (which would likely lead to large ℓ_∞ error over the set of answers). However, we posit two potentially superior alternatives whose performance we will investigate.

1. One alternative is to collect a large group of queries, and then use a state-of-the-art DP query answering mechanism to answer them all simultaneously. This is an example of answering queries in the "prespecified queries" setting (studied in Sections 3 and 4). With careful DP mechanism design or selection, this alternative typically leads to lower ℓ_∞ error over the set of answers than answering each query independently.

2. A separate alternative is along the lines of synthetic data generation, and is applicable to the Census Bureau if queries which have been posed in the past are in some sense similar to queries which analysts will likely pose in the future. Concretely, we hypothesize that the Census Bureau can leverage those past queries in conjunction with a state-of-the-art DP synthetic data generation mechanism to privately generate a synthetic dataset. Researchers can then pose their own queries directly against the synthetic dataset without needing to go through the Census Bureau, and without needing to worry about the original ACS respondents' privacy. This is an example of answering queries in the "partial knowledge" setting (studied in Section 5), as knowledge from the past is being used to inform the future. If the queries posed in the past are indeed similar to the queries posed in the future, then a synthetic dataset generated using the past queries has the potential to answer the future queries with low ℓ_∞ error.

1.1 Prior Work on Large-Scale Query Answering

To address answering a large number of queries under differential privacy in an improved manner over the naive interactive approach, two separate lines of research previously emerged: synthetic data generation, and workload evaluation. We describe both lines of research, then briefly introduce the state-of-the-art mechanism which we build upon in this work.

Synthetic Data Generation: One line of research studies the problem of answering a large number of queries via private synthetic dataset generation. In differentially private synthetic dataset generation, a DP mechanism is applied to the original, sensitive data in order to generate a synthetic dataset. The synthetic dataset's purpose is then to directly answer arbitrary queries posed in the future, without the further need to account for potential privacy leakage or manage differential privacy budgets. In this setting, aside from knowing the general query class, *no knowledge is typically assumed about which specific queries will be posed in the future.* The proven advantage of this approach is that DP synthetic datasets are theoretically capable of accurately answering an exponentially larger number of queries relative to the aforementioned interactive approach [GRU12, CKKL12, HRS12, GHRU13]. However, actually generating a synthetic dataset which accurately answers exponentially many queries has been proven intractable [DNR+09, UV11, Ull16], even for simple subclasses of statistical queries (e.g., 2-way marginals). Thus, a significant recent research focus has been on designing efficient mechanisms for privately generating synthetic datasets which accurately answer increasingly large numbers of queries [GAH+14, MSM19, VTB+20, LVW21].

Workload Evaluation: A separate line of research focuses on the problem of answering a large number of queries when the concrete query workload is prespecified; i.e., *when all queries are known in advance*. Pre-specifying the query workload enables researchers to design DP mechanisms to take advantage of the workload's structure in order to answer the queries with lower error relative to the interactive approach or the private synthetic dataset approach. Early research in this setting produced mechanisms with optimal or near-optimal error guarantees, but with impractical (typically exponential) running times for even modestly sized real-world problems [HR10, HLM12, GRU12, LMH+15]. As a result, recent research has focused on designing computationally efficient mechanisms to answer prespecified workloads with low error on real-world datasets [MMHM18, SS18, ABK+21], at the cost of losing the strong theoretical utility guarantees of prior works and thus necessitating thorough empirical utility evaluations to demonstrate their value.

***Relaxed Adaptive Projection* Mechanism:** Our approach for evaluating suitable (i.e., efficient and accurate) mechanisms in both our settings of interest builds on Aydore et al.'s [ABK+21] recently introduced *Relaxed Adaptive Projection* (`RAP`) mechanism. `RAP` is the current state-of-the-art mechanism for answering large sets of statistical queries in the setting where the query workload is prespecified. At a high-level, `RAP` works by:

1. Initializing a synthetic dataset D' in a relaxed data space (e.g., by relaxing a binary feature in the original dataset to the interval $[0, 1]$ in the synthetic dataset).

2. For each original prespecified query, specifying a surrogate query which is equivalent to the original in the unrelaxed data space, but which is differentiable everywhere in the relaxed space.

3. Iteratively applying an *Adaptive Selection* (`AS`) step followed by a *Relaxed Projection* (`RP`) step. In the `AS` step, adaptivity is introduced to allow the subset of queries with the highest error on D' to be privately selected. In the `RP` step, these selected queries' surrogates are used to optimize D' using standard gradient-based optimization techniques.

4. Finally, answering the original set of queries using the optimized synthetic dataset D'.

For k-way marginals, a canonical subclass of statistical queries [BCD+07, TUV12, GHRU13, CTUW14, CKS18] (formally defined in Section 2), Aydore et al. theoretically and empirically demonstrate that `RAP` outperforms prior state-of-the-art mechanisms. Theoretically, they provide an "oracle efficient" (i.e., assuming the optimization procedure achieves a global minima) utility result characterizing `RAP`'s error, showing that `RAP` achieves strictly lower error than the previous practical state-of-the-art mechanism [VTB+20]. Experimentally, they compare the `RAP` mechanism with prior state-of-the-art mechanisms [MSM19, VTB+20], demonstrating that `RAP` answers prespecified sets of queries with lower error.

1.2 Our Contributions

To answer this work's high-level research question, we make the following contributions in both settings of interest. In the classic setting where all queries are known in advance, our contributions are as follows.

- We overcome memory hurdles in `RAP`'s initial implementation by reimplementing `RAP` in a memory-efficient way, thus enabling the evaluation of significantly larger query spaces than previously considered.

- We utilize the new implementation to enhance `RAP`'s evaluation, evaluating `RAP` on larger query spaces (answering approximately 50x more queries) than in its initial evaluation, and conclusively determining the role that adaptivity from the `AS` step plays in `RAP`'s utility.

- We extend `RAP`'s applicability by expanding the class of queries that it evaluates, finding that it can efficiently and effectively answer more complex query classes than previously considered.

As a realistic intermediate setting that lies between the two classic extremes of no-knowledge vs. full-knowledge of which queries will be posed, we propose a new setting where partial knowledge of the future queries is available. In this new setting, our contributions are as follows.

- We concretely define this setting as well as how to measure utility within it. Specifically, we assume that a set of historical queries was independently drawn from some unknown distribution $\mathcal{T}_H$, and that the mechanism has access to these historical queries. In the future, the mechanism will be posed an arbitrary number of queries sampled from a distribution $\mathcal{T}_F$, which may be related to $\mathcal{T}_H$. We define the utility of the mechanism in terms of its generalization error; i.e., its expected error across these future queries drawn from $\mathcal{T}_F$ having been given access to the historical queries from $\mathcal{T}_H$.
- We assess how suitable RAP is for this new setting by formulating query distributions according to real-world phenomena, then empirically evaluating RAP's generalization error on these distributions. When future queries are drawn from the same distribution as the historical queries that RAP used to learn its synthetic dataset (i.e., $\mathcal{T}_H = \mathcal{T}_F$), we find that regardless of what the distribution is, RAP is able to achieve high utility. When the distribution of future queries diverges from the distribution of historical queries, we find that RAP's utility slowly and gracefully declines.

These contributions, in both the prespecified queries setting and the partial knowledge setting, definitively demonstrate the practical value of RAP and improve RAP's adoptability for real-world uses.

The remainder of this work is structured as follows. Beginning in Section 2, we provide a comprehensive overview of the relevant technical terminology and definitions, and detail the RAP mechanism that we build upon. In Section 3, we perform a focused but thorough reproducibility study on Aydore et al.'s [ABK+21] evaluation of the RAP mechanism. To accomplish this, we first improve RAP's implementation from the ground up, and then leverage the new implementation to enhance RAP's initial evaluation in order to strengthen our comprehension of its utility. Building on the improved RAP implementation, in Section 4 we expand the class of queries that RAP is able to accommodate. We then empirically evaluate RAP on this new class of queries, finding that it is able to efficiently answer large numbers of queries from this class while maintaining high utility. In Section 5, we concretely define our newly proposed setting where a mechanism is given partial knowledge of the queries that will be posed in the future. We define how we assess RAP's performance in this setting, and detail the distinct new ways that RAP's performance may be affected in this new setting. We then empirically evaluate RAP in this setting, finding that even with only partial knowledge of which queries will be posed in the future, RAP is able to efficiently and effectively achieve high utility. Finally, in Section 6, in addition to the related works already discussed in this section, we describe other important relevant works and the future directions they motivate related to this work.

2 Technical Preliminaries

In this section, we define the requisite technical terminology. The fundamental concepts introduced here were primarily presented in prior works [GAH+14, VTB+20, ABK+21]. We restate them to aid in understanding and contextualizing Aydore et al.'s RAP mechanism, which we use to answer this work's research questions. Towards this, we first define statistical queries and their subclasses that are relevant to this work. We then define what it means to be a "surrogate" query for one of these statistical queries. Next, we describe what workloads are and how we use them. Finally, we detail the RAP mechanism that we build on in this work. Because this work is notationally dense, Table 1 serves as a reference for the various symbols that we define.

2.1 Statistical Queries and their Subclasses

The general class of queries that we are interested in (which the RAP mechanism can, in theory, be used to answer) are statistical queries.

Definition 2.1 (Statistical query)**.** A *statistical query* q_ϕ is parameterized by a predicate $\phi : \mathcal{X} \to \{0, 1\}$; i.e., the predicate takes as input a record x of a dataset D, and outputs a boolean value. The statistical query is then defined as the normalized count of the predicate over all n records of the input dataset; i.e.,

$$q_\phi(D) = \frac{\sum_{x \in D} \phi(x)}{n}.$$

	Symbol	Usage
	ϵ, δ	Differential privacy parameters.
	$\mathcal{X},\ d,\ \mathcal{X}_i$	Data space $\mathcal{X}$ for any possible record consisting of d features. $\mathcal{X}_i$ is the domain of feature i.
	$D,\ n$	Dataset D containing n records from $\mathcal{X}$.
	q_ϕ	Statistical query q defined by the mean of the predicate ϕ over a set of records from $\mathcal{X}$.
	$Q,\ m,\ a$	Q is a vector of m queries, and a represents the answers to the vector of queries over the dataset D such that $Q(D) = a = (a_1, \dots, a_m)$.
	W	Threshold workload W which defines the concrete query vector Q.
	$q_{\phi_{S,y,k}}$	k-way marginal query specified by set S of k features and values y for each feature.
	$q_{\phi_{S,y,1}}$	1-of-k threshold query specified by set S of k features and values y for each feature.
$\star$	$q_{\phi_{S,y,r}}$	r-of-k threshold query specified by set S of k features and values y for each feature, and threshold r.
	$\mathcal{Y},\ d'$	Data space $\mathcal{Y}$ consisting of d' features, which is a relaxation of the one-hot encoded $\mathcal{X}$ data space.
	$D',\ n'$	Synthetic dataset D' containing n' features from $\mathcal{Y}$.
	$\hat{q}_{\hat{\phi}}$	Surrogate query $\hat{q}$ defined by the mean of the function $\hat{\phi}$ over a set of records from $\mathcal{Y}$.
	$\hat{Q}$	Vector of surrogate queries.
	$\hat{q}_{\hat{\phi}_T}$	Product query, specified by a set of features T.
$\star$	$\hat{q}_{\hat{\phi}_{T_+,T_-}}$	Generalized product query, specified by a set of positive and negated features T_+ and T_-.
$\star$	$\hat{q}_{\hat{\phi}_{T,r}}$	Polynomial threshold query, specified by a set of features T and integer r.
$\star$	err_P	Measure of a mechanism's present error, used when all queries are known in advance.
$\star$	err_F	Measure of mechanism's future error, used when only partial knowledge of queries is available in advance.
$\star$	$\mathcal{F}, \mathcal{T}$	Distribution $\mathcal{T}$ from which thresholds in a random workload are sampled i.i.d. in order to form a corresponding vector of consistent queries. The threshold distribution may be formed by a distribution over features $\mathcal{F}$.
	`RAP`, `AS`, `RP`	Relaxed Adaptive Projection mechanism, with its primary subcomponents: the Adaptive Selection and Relaxed Projection mechanisms.
	`RNM`	Report Noisy Max mechanism, used by the `AS` mechanism to select high-error queries.
	`GM`	Gaussian noise-addition mechanism, used as both a baseline mechanism as well as a subcomponent of `RAP` to privately answer queries directly.
$\star$	`OSAS`	Oneshot Adaptive Selection mechanism, introduced as more efficient a drop-in replacement for `RAP`'s `AS` mechanism.
	`All-0`	Baseline mechanism that returns only `0` for all queries.

Table 1: Comprehensive list of notation. Lines marked with a $\star$ indicate new concepts not found in [ABK+21].

Given a vector of m statistical queries Q, we define $Q(D) = (a_1, \ldots, a_m)$ to be the answers to each of the queries on D; i.e., $a_i = q_{\phi_i}(D)$ for all $i \in [m]$.

We now formally define the specific subclasses of statistical queries that we reference throughout this work. Let the space for each record in the dataset consist of d categorical features $\mathcal{X} = (\mathcal{X}_1 \times \cdots \times \mathcal{X}_d)$, where each $\mathcal{X}_i$ is the discrete domain of feature i, and let $x_i \in \mathcal{X}_i$ denote the value of feature i of record $x \in \mathcal{X}$. Prior works have primarily evaluated the subclass of statistical queries known as k-way marginals (also known as k-way contingency tables or k-way conjunctions) [BCD+07, TUV12, GHRU13, CTUW14, CKS18, MSM19, VTB+20], and typically focused specifically on 3-way and 5-way marginals.

Definition 2.2 (k-way marginal)**.** A *k-way marginal query* $q_{\phi_{S,y,k}}$ is a statistical query whose predicate $\phi_{S,y,k}$ is specified by a set S of k features $f_1 \neq \cdots \neq f_k \in [d]$ and a target $y \in (\mathcal{X}_{f_1} \times \cdots \times \mathcal{X}_{f_k})$, given by

$$\phi_{S,y,k}(x) = \begin{cases} 1 & \text{if } x_{f_1} = y_1 \ \wedge \cdots \ \wedge x_{f_k} = y_k \\ 0 & \text{otherwise.} \end{cases}$$

Informally, a row satisfies the predicate if *all* of its values match the target on the specified features. A *k-way marginal* is then specified by a set S of k features, and consists of all $(\Pi_{i=1}^{k} |\mathcal{X}_{f_i}|)$ k-way marginal queries with feature set S.

1-of-k thresholds (also known as k-way disjunctions) were briefly evaluated in [ABK+21], and are defined similarly.

Definition 2.3 (1-of-k threshold)**.** A *1-of-k threshold query* $q_{\phi_{S,y,1}}$ is a statistical query whose predicate $\phi_{S,y,1}$ is specified by a set S of k features $f_1 \neq \cdots \neq f_k \in [d]$ and a target $y \in (\mathcal{X}_{f_1} \times \cdots \times \mathcal{X}_{f_k})$, given by

$$\phi_{S,y,1}(x) = \begin{cases} 1 & \text{if } x_{f_1} = y_1 \ \vee \cdots \ \vee x_{f_k} = y_k \\ 0 & \text{otherwise.} \end{cases}$$

Informally, a row satisfies the predicate if *any* of its values match the target on the specified features. A *1-of-k threshold* is then specified by a set S of k features, and consists of all $(\Pi_{i=1}^{k} |\mathcal{X}_{f_i}|)$ 1-of-k threshold queries with feature set S.

Finally, in this work, we evaluate a generalization of both of these subclasses of statistical queries: r-of-k thresholds [KLPV87, Lit88, HW04, TUV12, Ull13, ABK+21].

Definition 2.4 (r-of-k threshold)**.** An *r-of-k threshold query* $q_{\phi_{S,y,r}}$ is a statistical query whose predicate $\phi_{S,y,r}$ is specified by a positive integer $r \leq k$, a set S of k features $f_1 \neq \cdots \neq f_k \in [d]$, and a target $y \in (\mathcal{X}_{f_1} \times \cdots \times \mathcal{X}_{f_k})$. The predicate is then given by

$$\phi_{S,y,r}(x) = \mathbb{1}\left[\sum_{i=1}^{k} \mathbb{1}[x_{f_i} = y_i] \geq r\right].$$

Informally, a row satisfies the predicate if *at least* r of its values match the target on the specified features. An *r-of-k threshold* is then specified by positive integer $r \leq k$ and a set S of k features, and consists of all $(\Pi_{i=1}^{k} |\mathcal{X}_{f_i}|)$ r-of-k threshold queries with feature set S. This class generalizes k-way marginals when $r = k$, and generalizes 1-of-k thresholds when $r = 1$.

The expressiveness of r-of-k thresholds make them more useful than k-way marginals, as they enable more nuanced queries to be easily and intuitively posed. This is particularly useful when the implications behind categories of distinct features in a dataset have some overlap. For instance, in the motivating U.S. Census example, there were several features with categories that were indicative of a substandard quality of living. Requiring someone to belong to *all* of the categories (as a k-way marginal requires) is overly restrictive, and r-of-k thresholds allow this restrictiveness to be relaxed.

Remark. We say that any r-of-k threshold query (and, by extension, any k-way marginal query or 1-of-k threshold query) specified by r, k, S, and y is *consistent* with the r-of-k threshold specified by r, k, and S. That is, we often refer to an r-of-k threshold simply as the features it specifies, whereas a query *consistent with* that r-of-k threshold is one which specifies concrete target values corresponding to those features.

2.2 Surrogate Queries

Aydore et al. [ABK+21] introduce surrogate queries to replace the original statistical queries with queries that are similar, but that are amenable to first-order optimization methods. These first-order optimization methods, thanks to significant recent advances in hardware and software tooling, can enable highly efficient learning of synthetic datasets.

Definition 2.5 (Surrogate Query)**.** A *surrogate query* $\hat{q}_{\hat{\phi}}$ is parameterized by function $\hat{\phi} : \mathcal{Y} \to \mathbb{R}$; i.e., the function takes as input a record $x \in \mathcal{Y}$ from a dataset D', and outputs a real value. The surrogate query is then defined as the normalized count of the function over all n' records of the input dataset; i.e.,

$$\hat{q}_{\hat{\phi}}(D') = \frac{\sum_{x \in D'} \hat{\phi}(x)}{n'}.$$

The only distinctions between the definitions of a surrogate query with $\hat{\phi}$ and a statistical query with ϕ are that $\hat{\phi}$'s domain may be different than ϕ's, and $\hat{\phi}$'s codomain is the entire real line instead of $\{0, 1\}$.

We are interested in surrogate queries that are *equivalent extended differentiable queries* (EEDQs) as defined in [ABK+21].

Definition 2.6 (Equivalent Extended Differentiable Query)**.** Let q_ϕ be an arbitrary statistical query parameterized by $\phi(x) : \mathcal{X} \to \{0, 1\}$, and let $\hat{q}_{\hat{\phi}}$ be a surrogate query parameterized by $\hat{\phi} : \mathcal{Y} \to \mathbb{R}$. We say that $\hat{q}_{\hat{\phi}}$ is an *equivalent extended differentiable query* to q_ϕ if it satisfies the following properties:

1. $\hat{\phi}$ is differentiable over $\mathcal{Y}$. I.e., for every $x \in \mathcal{Y}$, $\nabla\hat{\phi}(x)$ is defined.

2. $\hat{\phi}$ agrees with ϕ on every possible database record that results from a one-hot encoding. I.e., for every $x \in \mathcal{X}$ where $h(x)$ represents a one-hot encoding[*] of x: $\phi(x) = \hat{\phi}(h(x))$.

Notation of Feature Spaces: Recall the original feature space $\mathcal{X} = (\mathcal{X}_1 \times \cdots \times \mathcal{X}_d)$, where each $\mathcal{X}_i$ is the discrete domain of feature i, and let t_i be the number of distinct values/categories that $\mathcal{X}_i$ can attain. A one-hot encoding $h(x)$ of any record x results in a binary vector $\{0, 1\}^{d'}$, where $d' = \sum_{i=1}^{d} t_i$. Just as in [ABK+21], we are interested in constructing a synthetic dataset that lies in a continuous relaxation of this binary feature space. A natural relaxation of $\{0, 1\}^{d'}$ is $[0, 1]^{d'}$, so we adopt $\mathcal{Y} = [0, 1]^{d'}$ as the relaxed space for the remainder of this work.

As an illustrative example of an EEDQ, we define the class of EEDQ's used by Aydore et al. for k-way marginals. Concretely, [ABK+21] defines the class of surrogate queries known as *product queries*, and shows how to construct an EEDQ product query for any given k-way marginal.

Definition 2.7 (Product Query)**.** Given a subset of features $T \subseteq [d']$, the *product query* $\hat{q}_{\hat{\phi}_T}$ is a surrogate query parameterized by function $\hat{\phi}_T$ which is defined as $\hat{\phi}_T(x) = \prod_{i \in T} x_i$.

Lemma 2.8 ([ABK+21], Lemma 3.3)**.** Every k-way marginal query $q_{\phi_{S,y,k}}$ has an EEDQ in the class of product queries. By construction, every $\hat{\phi}_T$ satisfies the requirement that it is defined over the entire relaxed space $\mathcal{Y}$ and is differentiable. Additionally, for every $q_{\phi_{S,y,k}}$, there is a corresponding product query $\hat{q}_{\hat{\phi}_T}$ with $|T| = k$ such that for every $x \in \mathcal{X} : \phi_{S,y,k}(x) = \hat{\phi}_T(h(x))$. We construct this T in the following straightforward way: for every $i \in S$, we include in T the coordinate corresponding to $y_i \in \mathcal{X}_{f_i}$.

[*]A one-hot encoding of a categorical feature $\mathcal{X}_i$ with t_i categories is a mapping from each category to a unique $1 \times t_i$ binary vector that has exactly 1 non-zero coordinate.

2.3 Threshold Workloads

It was standard in prior works to evaluate *workloads* of k-way marginals [LMH+15, MMHM18, MSM19, VTB+20, LVS+21, LVW21]. A k-way marginal workload W is specified by a set of k-way marginals, $W = \{S_1, \ldots, S_{|W|}\}$ such that each $S_i \in W$ is a set of k features. This workload W defines a concrete query vector Q which consists of all queries consistent with each marginal in W. Since Q is defined by the marginal workload defines, Q is commonly referred to as the *query workload.* For example, a workload may be specified by the following two 3-way marginals, $W = \{(1, 2, 5), (2, 3, 7)\}$, and would therefore define the query vector Q containing all marginal queries consistent with those feature sets. The number of queries in this query vector would then be $|Q| = |\mathcal{X}_1||\mathcal{X}_2||\mathcal{X}_5| + |\mathcal{X}_2||\mathcal{X}_3||\mathcal{X}_7|$.

Since our work extends the class of queries from marginals to r-of-k thresholds, rather than a workload being specified by a set of marginals, we say that a workload W is specified by a set of r-of-k thresholds. W similarly defines the concrete query vector Q which consists of all r-of-k threshold queries consistent with each r-of-k threshold in W. For example, when $r = 1$ and $k = 3$, we can specify a similar workload as before $W = \{(1, 2, 5), (2, 3, 7)\}$ which defines query workload Q containing the same number of consistent queries as before ($|Q| = |\mathcal{X}_1||\mathcal{X}_2||\mathcal{X}_5| + |\mathcal{X}_2||\mathcal{X}_3||\mathcal{X}_7|$) – however, here each $q \in Q$ is a 1-of-3 threshold query instead of a 3-way marginal query.

Lastly, we let $\hat{Q}$ denote the corresponding vector of surrogate queries for Q. We use threshold workloads (and their corresponding vector of all consistent queries) for the empirical evaluations of our mechanisms.

2.4 *Relaxed Adaptive Projection* (RAP) Mechanism

We now describe the details of the RAP mechanism, including how it works as well as its DP guarantee.

Algorithm 1 formally defines the RAP mechanism. The input to the mechanism is the dataset D of sensitive user data, the desired size of the synthetic dataset n', privacy parameters (ϵ, δ), a vector of m statistical queries Q and their corresponding surrogate queries $\hat{Q}$, adaptiveness parameters $T, K \in [m]$. The final outputs are (1) an n'-row synthetic dataset, and (2) estimates to the original queries Q obtained by evaluating their surrogate queries on the synthetic dataset; i.e., RAP outputs (1) D' and (2) $\hat{Q}(D')$.

Non-Adaptive Case: In its most basic form ($T = 1, K = m$), RAP employs no adaptivity. Here, the vector of m queries are first privately answered directly on the sensitive dataset D using the *Gaussian Mechanism* (GM). These answers, along with the vector of surrogate queries $\hat{Q}$ and a uniformly randomly initialized n'-row synthetic dataset D', are passed to the *Relaxed Projection* mechanism (RP, Algorithm 3). The RP subcomponent utilizes an iterative gradient-based optimization procedure (such as SGD) to update D' by minimizing the disparity between the surrogate queries answers on D' and the privatized answers on the sensitive dataset D. After iterative update, the Sparsemax transformation is applied to every feature encoding in each row of D'. Once the procedure reaches a stopping condition (e.g., $\hat{Q}(D')$ is within a certain tolerance of $\tilde{a}$, or a certain number of iterations have occurred), RP returns the final D'. RAP then returns D' along with estimated answers to the query workload $\hat{Q}(D')$.

Adaptive Case: In the more general case, RAP proceeds in $T > 1$ rounds. In each round t, RAP uses the *Adaptive Selection* (AS) mechanism to select K new queries to add to the set Q_s. AS iteratively uses the Gumbel noise *Report Noisy Max* (RNM) [CCK+16, DR19] and GM mechanisms together to privately choose the K queries that have the largest disparity between their current answers on the synthetic dataset D' and their answers on the true dataset D. The RP mechanism is then applied only to this subset Q_s containing tK queries in each round, rather than applying RP in 1 round on the full vector of privately answered queries Q (as in the non-adaptive case). Aydore et al. claim that the aim of incorporating this adaptivity is to expend the privacy budget more wisely by selectively answering only the $TK \ll m$ total worst-performing queries.

Concentrated Differential Privacy

To state and understand RAP's DP guarantee, we must briefly discuss *zero-concentrated differential privacy* (zCDP) [BS16].

Although RAP is given ϵ and δ values as input and in turn guarantees (ϵ, δ)-DP, its DP sub-mechanisms and corresponding privacy proof are in terms of ρ-zCDP. Zero-concentrated differential privacy is a different definition

Algorithm 1 Relaxed Adaptive Projection (RAP) Mechanism

Input

- D: Dataset of n records from space $\mathcal{X}$.
- $Q, \hat{Q}$: A vector of m statistical queries and their corresponding surrogate queries.
- $n', \mathcal{Y}$: Desired size of synthetic dataset with records from relaxed space $\mathcal{Y}$.
- T: Number of rounds of adaptiveness.
- K: Number of queries to select per round of adaptiveness.
- ϵ, δ: Differential privacy parameters.

Body

```
1:  Let $\rho = \epsilon + 2\left(\log(\frac{1}{\delta}) - \sqrt{\log(\frac{1}{\delta})(\epsilon + \log(\frac{1}{\delta}))}\right)$.
2:  Independently uniformly randomly initialize $D' \in \mathcal{Y}^{n'}$.
3:  if $T = 1,\ K = m$ then
4:      for $i = 1, 2, \ldots, m$ do
5:          Let $\tilde{a}_i = \texttt{GM}(D, q_i, \rho/m)$.
6:      end for
7:      Let $D' = \texttt{RP}(\hat{Q}, \tilde{a}, D')$.
8:  else
9:      Let $Q_s = \emptyset$.
10:     for $t = 1, 2, \ldots, T$ do
11:         Let $Q_s, \tilde{a} = \texttt{AS}(D, D', Q, \hat{Q}, Q_s, K, \frac{\rho}{T})$.
12:         Let $\hat{Q}_s = (\hat{q}_i : q_i \in Q_s)$.
13:         Let $D' = \texttt{RP}(D', \hat{Q}_s, \tilde{a})$.
14:     end for
15: end if
16: Return: Final synthetic dataset $D'$ and answers $\hat{Q}(D')$.
```

Algorithm 2 Adaptive Selection (AS) Mechanism

Input

- D, D': Dataset of n records from space $\mathcal{X}$, and synthetic dataset of n' records from relaxed space $\mathcal{Y}$.
- $Q, \hat{Q}$: Vector of all statistical queries and their corresponding surrogate queries.
- Q_s: Set of already selected queries.
- K: Number of new queries to select.
- ρ: Differential privacy parameter.

Body

```
1: for $j = 1, 2, \ldots, K$ do
2:     Let $\Delta = (|\hat{q}_i(D) - \hat{q}_i(D')| : q_i \in Q \setminus Q_s)$.
3:     Let $i = \texttt{RNM}(\Delta, \frac{\rho}{2K})$
4:     Add $q_i$ into $Q_s$.
5:     Let $\tilde{a}_i = \texttt{GM}(D, q_i, \frac{\rho}{2K})$.
6: end for
7: Return: $Q_s$ and $\tilde{a} = (\tilde{a}_i : q_i \in Q_s)$.
```

Algorithm 3 Relaxed Projection (RP) Mechanism

```
Input
• $D'$: Synthetic dataset of $n'$ records from relaxed space $\mathcal{Y}$.
• $\hat{Q}$: Vector of surrogate queries.
• $\tilde{a}$: Vector of "true" privatized answers corresponding to each surrogate query.
Body
1: Use any iterative differentiable optimization technique (SGD, Adam, etc.) to attempt to find:
       $$D' = \arg\min_{D' \in \mathcal{Y}^{n'}} ||\hat{Q}(D') - \tilde{a}||_2^2,$$
   applying the Sparsemax transformation to every feature encoding in each row of $D'$ between each iteration.
2: Return: $D'$.
```

of DP that provides a weaker guarantee than pure DP but a stronger guarantee than approximate DP. It is formally defined as follows.

Definition 2.9 ([BS16]). A randomized mechanism $\mathcal{M}$ is ρ-zCDP if and only if for all neighboring input datasets D and D' that differ in precisely one individual's data and for all $\alpha \in (1, \infty)$, the following inequality is satisfied:

$$\mathbb{D}_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) \leq \rho\alpha,$$

where $\mathbb{D}_\alpha(\cdot||\cdot)$ is the α-Rényi divergence.

We omit a detailed discussion of zCDP in this work, referring an interested reader to Bun and Steinke's work [BS16] for more details. However, its value for RAP comes from the fact that zCDP has better composition properties than approximate DP, yet RAP's final composed zCDP guarantee (parameterized by ρ) can be converted back into an (ϵ, δ)-DP guarantee. This converted (ϵ, δ)-DP guarantee is better than if standard composition results of approximate DP had been directly applied.

We now informally state these composition and conversion properties. zCDP's composition property ensures that if two mechanisms satisfy ρ_1-zCDP and ρ_2-zCDP, then a mechanism that sequentially composes them satisfies ρ-zCDP with $\rho = \rho_1 + \rho_2$. zCDP's conversion property ensures that if a mechanism satisfies ρ-zCDP, then for any $\delta > 0$, the mechanism also satisfies (ϵ, δ)-DP with $\epsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$.

Finally, we define the two fundamental DP mechanisms used in RAP — GM and RNM — and state their DP guarantees in terms of zCDP. The first mechanism is the Gaussian mechanism, which we restate here in terms of zCDP and for the particular use case of answering a single statistical query.

Definition 2.10. The Gaussian mechanism $\texttt{GM}(D, q_i, \rho)$ takes as input a dataset $D \in \mathcal{X}^n$, a statistical query q_i, and a zCDP parameter ρ. It outputs $a_i = q_i(D) + Z$, where $Z \sim \text{Normal}(0, \sigma^2)$ and $\sigma^2 = \frac{1}{2n^2\rho}$.

Lemma 2.11 ([BS16]). For any query q_i and $\rho > 0$, the $\texttt{GM}(D, q_i, \rho)$ satisfies ρ-zCDP.

The second fundamental mechanism that RAP uses is the Gumbel noise Report Noisy Max (RNM) mechanism.

Definition 2.12. The Report Noisy Max mechanism $\texttt{RNM}(D, \Delta, \rho)$ takes as input a dataset $D \in \mathcal{X}^n$, a vector of real values Δ, and a zCDP parameter ρ. It outputs the index of the highest noisy value in Δ; i.e., $i^* = \arg\max_i \Delta_i + Z_i$, where each $Z_i \sim \text{Gumbel}\left(\frac{1}{\sqrt{2\rho|D|^2}}\right)$.

Lemma 2.13 ([DR19]). For any real vector Δ and $\rho > 0$, the $\texttt{RNM}(D, \Delta, \rho)$ satisfies ρ-zCDP.

With these fundamental mechanisms and their zCDP guarantees defined, we are now able to formally reproduce Aydore et al.'s original theorem and proof of RAP's DP guarantee.

Theorem 2.14 ([ABK+21])**.** For any class of queries and surrogate queries Q and $\hat{Q}$, and for any set of parameters n', T, and K, the RAP mechanism satisfies (ϵ, δ)-DP.

Proof. First, consider the non-adaptive case where $T = 1, K = m$. Here, the sensitive dataset D is only accessed via m invocations of the Gaussian mechanism, each with privacy ρ/m. Therefore, by the composition property of zCDP, RAP satisfies ρ-zCDP. Thus, by our choice of ρ in line 1, we conclude that RAP satisfies (ϵ, δ)-DP.

Next, assume $T > 1$. RAP executes T iterations of its loop, only accessing the sensitive dataset D via the Adaptive Selection (AS) mechanism each iteration. Thus, we seek to prove that the AS mechanism satisfies ρ/T-zCDP. Each invocation of the AS mechanism receives as input the privacy parameter $\rho' = \rho/T$, and accesses the sensitive dataset via K invocations of RNM and K invocations of GM. Each invocation of either mechanism ensures $\frac{\rho'}{2K}$-zCDP, and therefore by the composition property of zCDP, the total $2K$ mechanism invocations ensure ρ'-zCDP. Thus, the AS mechanism satisfies ρ/T-zCDP. Leveraging zCDP's composition property again, because RAP invokes AS T times, RAP therefore satisfies ρ-zCDP. Finally, by our choice of ρ in line 1, we conclude that RAP satisfies (ϵ, δ)-DP. □

3 Enhancing RAP's Evaluation

In this section, we address our first two contributions in the setting where all queries are prespecified: we strengthen and clarify our understanding of RAP's utility by performing a thorough reproducibility study on two important aspects of Aydore et al.'s evaluation of RAP. These two aspects are:

1. The benefit of RAP's adaptive component relative to its non-adaptive component was unclear in its initial evaluation. We conclusively determine and quantify this component's utility benefit, finding that it is crucial for enabling RAP to achieve high utility.

2. RAP was initially only evaluated on highly reduced portions of the query space. We instead evaluate RAP's utility across the entire query space, answering up to 50x more queries than in its initial evaluation.

The first aspect is significant because it improves our understanding of how RAP's adaptivity parameters affect its utility and establishes whether RAP's adaptive component is necessary in order to achieve high utility. The second aspect is important because RAP's initial evaluation on highly reduced portions of the query space yielded potentially biased utility results. By instead evaluating RAP across the entire query space, we establish RAP's unbiased utility and determine what impact reducing the query space has on RAP's utility. In order to evaluate both aspects, we must reimplement RAP from the ground up in order to improve its efficiency for evaluating large sets of prespecified queries. We then use the new implementation to evaluate both aspects, clarifying the value of the RAP mechanism and thus improving its adoptability for practical uses.

To make the description of our improved evaluation precise, in Section 3.1 we define the utility metric used by Aydore el al. and by the prior state-of-the-art mechanisms for answering prespecified queries, which we also use in our evaluations. We then discuss in Section 3.2 the details and implications of the two aspects of Aydore et al.'s initial evaluation of RAP that we are improving upon. In Section 3.3, we detail the particular obstacle in RAP's initial implementation which prevents its use for our improved evaluation. To overcome this obstacle, we reimplement RAP from the ground up and make its implementation publicly available[†]. Finally, in Section 3.4, we describe how we use our improved implementation to perform our enhanced evaluation of RAP.

With regards to the role of adaptivity in RAP, we not only find that it is crucial to achieving high-utility, we also quantitatively and definitively measure how RAP's adaptivity parameters (T and K) affect its utility. This motivates new, more efficient search strategies to find optimal T and K values, thus reducing RAP's computational burden and privacy cost in practice. With regards to evaluating RAP on the full query space, we find that Aydore et al.'s initial evaluation of RAP on a reduced portion of the query space likely *underestimated* RAP's utility. This was due to their reduced query space having less "sparsity" in the query answers (i.e., a larger portion of the queries they evaluated had non-0 answers). This finding motivates a new line of research on mechanisms for the separate cases of when query answers are and are not sparse. Together, the improved RAP implementation combined with the enhanced evaluation clarifies the value of the RAP mechanism, and thus improves RAP's adoptability and usability in practice.

[†]https://github.com/bavent/large-scale-query-answering.

3.1 Measuring Utility of Prespecified Queries

We define the concrete utility measure used in prior works to evaluate DP mechanisms that answer prespecified sets of statistical queries. Prior works in this setting measured the utility of DP mechanisms in terms of a mechanism's maximum error over the answers to all queries in the prespecified query set [MSM19, VTB+20, LVW21, ABK+21]. We refer to this measure of utility as *present utility*, since it is the error on the set of presently available queries, and measure it in terms of the negative of *present error*; i.e., a mechanism with low present error has high present utility, and vice versa. This error measure is formally defined as follows.

Definition 3.1 (Present error)**.** Let $a = Q(D) = (a_1, \dots, a_m)$ be the true answers to a given query vector Q on dataset D, and let $\tilde{a} = (\tilde{a}_1, \dots, \tilde{a}_m)$ be mechanism M's corresponding answers to the query vector. Then err_P is the present error of the mechanism, defined as $\mathrm{err}_P(M, D, Q) = \mathbb{E}_{M(D)} \|a - \tilde{a}\|_\infty$, where the expectation is over the randomness of the mechanism.

We choose the ℓ_∞ norm as the base metric for present error because of its use in Aydore et al.'s evaluation of RAP and because it is the most popular norm utilized in the most closely related literature [MSM19, VTB+20, LVW21, ABK+21]. However, other norms (e.g., ℓ_1 and ℓ_2) and even definitions of error may be equally valid in the prespecified queries setting depending on the practical use case [TMH+21]. Thus, although we do not empirically evaluate RAP on such alternative definitions, investigating how the findings in this work change based on the error definition is an excellent direction for future work.

3.2 Focus of RAP's Reevaluation

We now detail the two primary aspects of Aydore et al.'s evaluation of RAP that we enhance in this work, and how their origins trace back to a particular challenge in RAP's initial implementation.

Adaptivity Evaluation: The first aspect that we address in RAP's reevaluation is how RAP's adaptive component affects its utility. To provide context, we briefly describe the non-adaptive form of RAP. We then describe the adaptive form of RAP and the motivation behind its design. Finally, we detail how Aydore et al.'s evaluation of RAP omitted studying the adaptive component's effect on utility, and we describe why that is an issue.

In its non-adaptive form, the RAP mechanism essentially reduces to privately answering the full query vector Q with the Gaussian Mechanism, then applying the RP mechanism to generate a synthetic dataset. This non-adaptive form of the RAP mechanism is a novel reimagining of the classic *Projection Mechanism* [NTZ13], a near-optimal but computationally intractable mechanism for answering prespecified queries. By leveraging a relaxation of the query space and utilizing EEDQs, Aydore et al. describe how their non-adaptive RAP mechanism can use modern tools (e.g., GPU-accelerated optimization) to efficiently generate a relaxed synthetic dataset which can hypothetically answer the prespecified queries with low (albeit non-optimal) error. Moreover, they prove a theoretical result (Theorem 4.1, [ABK+21]) which confirms the power of the non-adaptive RAP mechanism, achieving a $\sqrt{d'}$ factor of utility improvement over the prior state-of-the-art mechanism.

Aydore et al. go on to describe the full adaptive form of RAP parameterized by T and K. This adaptive form of RAP optimizes the synthetic dataset iteratively over T separate rounds, in each round adaptively selecting K new queries to incorporate into the optimization procedure. Their stated motivation for introducing adaptivity into RAP was to more wisely expend the privacy budget by adaptively optimizing over a small number of "hard" queries, and they conjecture (without a result similar to that of their Theorem 4.1) that such adaptivity will result in higher utility than that achieved by the non-adaptive form of RAP.

Aydore et al. then perform an empirical evaluation of RAP across a range of parameters and datasets, and establish that it achieves state-of-the-art utility — however, the utility benefits of RAP's adaptivity are left unanalyzed. Specifically, in all evaluations they report the best utility of RAP across $2 \le T \le 50$ and $5 \le K \le 100$. There are two issues related to this.

1. The values of T and K that achieved the maximum utility are not reported, only what that maximum utility was. Thus, it is unclear how these parameters affect utility. This is problematic in practice because not only is evaluating RAP on multiple choices of T and K computationally expensive, but because each evaluation consumes a portion of the overall differential privacy budget.

2. The non-adaptive form of RAP is not empirically evaluated. Without evaluating the non-adaptive RAP mechanism as a baseline, there is no meaningful way to understand or measure the benefit of adaptivity.

Combined, these two issues leave open the question of how valuable the adaptive component of RAP is, and to what extent its adaptivity affects utility.

Query Space Evaluation: The second aspect that we address in RAP's reevaluation is how reducing the query space affects RAP's utility for answering k-way marginals. To begin, we describe the motivation behind evaluating this aspect: that for computational ease, Aydore et al. only evaluated RAP on a reduced portion of the query space. We then detail how this reduction may have biased their evaluation's results.

Aydore et al.'s empirical evaluation focuses on RAP's utility for answering k-way marginals, specifically 3-way and 5-way marginals. Reviewing the code of their published RAP implementation, we determined that a heuristic filtering criterion of the query space was being applied to remove any "large" marginals from possible evaluation. Specifically, any marginal which had more consistent queries than the number of records in the dataset (n) was not considered for evaluation. The impact that filtering had on the evaluated workloads varied depending on k and n. For instance, with 3-way marginals on the ADULT dataset, the filtering criterion removed the top 24% largest 3-way marginals which accounted for over 90% of all consistent queries. With 5-way marginals on the ADULT dataset, this filtering criterion removed the top 92% largest 5-way marginals which accounted for over 99.99% of all consistent queries.

Discussing this discrepancy directly with the authors[‡] revealed that the filtering criterion was an intentional choice meant to reduce the computational burden during experimentation, and they conjectured that removing this criterion and rerunning all experiments would yield results comparable to those obtained by increasing the workload size. Since all baseline mechanisms were evaluated on the same query vectors, the filtering criterion does not result in favorable utility for RAP relative to the prior state-of-the-art mechanisms that serve as their baselines. However, for marginals with a significantly larger number of consistent queries than n, most queries will evaluate to 0 by a Pigeonhole principle argument. Thus, the filtering criterion may result in favorable utility for RAP relative to the naive baseline mechanism that they consider in their work: `All-0`, the mechanism which outputs 0 as the answer to every query. This leaves open the question of RAP's utility on large, unfiltered query spaces, both in absolute terms and relative to the baseline `All-0` mechanism.

3.3 Reimplementing RAP

We now describe why these two aspects cannot be evaluated using Aydore et al.'s initial RAP implementation: briefly, the amount of memory required by the implementation is inordinate. We then detail how we overcome this challenge by reimplementing RAP in a way that trades-off a significant amount of memory usage for a potential increase in runtime.

Conceptually, both aspects could be evaluated using Aydore et al.'s published code. However, evaluating either the non-adaptive form of RAP or evaluating a larger portion of the query space both lead to the same obstacle: Aydore et al.'s RAP implementation requires an inordinate amount of memory to answer the corresponding large number of queries. We have identified several portions of their code where this memory bottleneck occurs, all of which fail to execute either when the total number of consistent queries is "too large" or when any marginal has "too many" consistent queries. Consequently, Aydore et al. were unable to evaluate either the non-adaptive form of RAP or a significant portion of the k-way marginals' consistent query space.

The high-level idea behind our approach for overcoming this implementation challenge is to trade-off some of RAP's required memory for a potential increase in its runtime. Our motivation for this approach is inspired by recent advances in differentially private deep learning literature. In particular, the canonical DP-SGD mechanism [ACG+16] for training machine learning models with differential privacy had been plagued by poor computational performance due to several of its underlying operations (e.g., per-example gradient clipping, uniformly random batch sampling without replacement, etc.) not being natively supported by modern machine learning frameworks. More recently however, several highly performant DP-SGD implementations [Pap19, YSS+21, SVK21] have been deployed which

[‡]https://github.com/amazon-research/relaxed-adaptive-projection/issues/2

dramatically decrease the mechanism's runtime in exchange for a mild increase in its memory usage. To our knowledge, our high-level approach is the first in DP literature to make practical use of this trade-off in the opposite direction: decreasing the mechanism's memory requirement by increasing its runtime.

Concretely, we overcome this implementation challenge by reimplementing RAP via the following high-level steps. First, we reduce the maximal memory requirement in RAP's original implementation caused by the original implementation's implicit evaluation all marginals (or, more generally, all thresholds) in parallel. We accomplish this by evaluating each marginal (or threshold) sequentially in order to distribute the computational burden. To further reduce the overall memory requirement, rather than explicitly enumerating and storing every query consistent with each marginal (threshold), we represent the queries implicitly and only convert a query to its explicit representation when it is needed for evaluation. To evaluate arbitrary sets of such individual queries, we implement the core EEDQ evaluation function from the ground up by designing a simple, direct function to efficiently evaluate arbitrary predicates. With such a function implemented, we then leverage a combination of powerful language features — namely vectorizing maps and just-in-time compilation in JAX [BFH+18] — to enable efficient evaluation, summation, and differentiation of large sets of predicates without exceeding memory constraints.

In addition to these implementation improvements which primarily serve to reduce RAP's memory requirement, we additionally incorporate an algorithmic improvement based on recent theoretical findings to help offset the increased runtime from our aforementioned deparallelization step. Specifically, by trivially adapting the *Oneshot Top-K Selection with Gumbel Noise* mechanism [DR19, CR21] to our setting, we replace RAP's iterative Adaptive Selection (AS) mechanism with the more efficient Oneshot Adaptive Selection (OSAS) mechanism in Alg. 4. The results of [DR19] prove that the OSAS mechanism is probabilistically equivalent to AS (i.e., both mechanisms have identical output distributions, and thus achieve identical privacy and utility), but OSAS requires only 1 pass over a set of values in order to select the top-K instead of the K passes that AS requires.

Algorithm 4 Oneshot Adaptive Selection (OSAS) Mechanism

```
Input
• $D, D'$: The dataset and synthetic dataset.
• $Q, \hat{Q}$: A vector of all statistical queries and their corresponding surrogate queries.
• $Q_s$: A set of already selected queries.
• $K$: The number of new queries to select $K$.
• $\rho$: Differential privacy parameter.

Body
1: Let $\Delta = (|\hat{q}_i(D) - \hat{q}_i(D')| : q_i \in Q \setminus Q_s)$.
2: Let $I$ denote the indices of the top-$K$ values of: $\Delta_i + Z_i$, where $Z_i \overset{\text{iid}}{\sim} \texttt{Gumbel}\left(\sqrt{\frac{K}{2\rho|D|^2}}\right)$.
3: Let $\tilde{a}_i = \texttt{GM}(D, q_i, \frac{\rho}{2K}) \quad \forall i \in I$.
4: Let $Q_s = Q_s \cup \{q_i\}_{i \in I}$.
5: **Return:** $Q_s$ and $\tilde{a} = (\tilde{a}_i : q_i \in Q_s)$.
```

Figure 1 compares our new implementation to Aydore et al.'s original implementation without filtering out any large marginals. Specifically, this figure shows the runtimes of both implementations executing the non-adaptive and adaptive variants of RAP given the same amount of GPU memory on two datasets across a range of workload sizes[§]. We find that for the non-adaptive variant of RAP, the original implementation was only able to evaluate tiny workloads, while our new reimplementation was able to evaluate massive workloads (albeit, with a very high runtime); this represents a 500x improvement in memory efficiency for our reimplementation. For the adaptive variant of RAP (specifically, with T=16 and K=4), we find the our reimplementation's runtime is comparable to the original implementation's — outperforming it slightly on one dataset, while being outperformed slightly on the other. On the ADULT dataset, both implementations were able to exhaustively evaluate the complete space of marginals.

[§]The runtimes for both implementations (and all subsequent evaluations in this work) were executed on an Nvidia RTX 3090 consumer GPU with 24 GB VRAM.

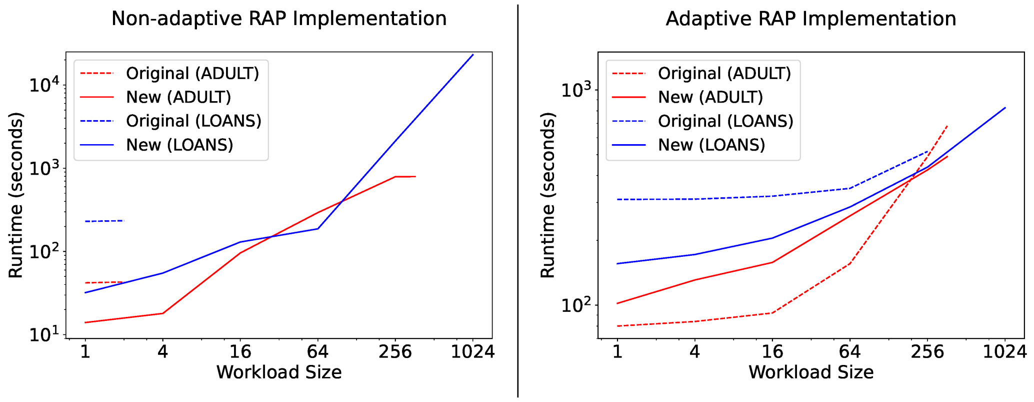

Figure 1: Runtime evaluations of non-adaptive and adaptive RAP variants on the original implementation and reimplementation, on both ADULT and LOANS datasets.

Dataset	Records	Features	Binarized Features
ADULT	48,842	14	588
LOANS	42,535	48	4,427

Table 2: Datasets for empirical evaluations. Binarized features represent the features after a transformation via one-hot encoding.

On the LOANS dataset, the original implementation was able to consistently evaluate marginal workloads of size 256, but was unable to consistently evaluate the largest workload size of 1024; this represents up to a 4x improvement in memory efficiency for our reimplementation.

3.4 Reevaluating RAP

Using our new implementation, we reevaluate both the adaptivity and query space aspects of RAP, enabling new findings. We start by simply establishing RAP's present utility for answering k-way marginals on unbiased random samples of the full marginal space (i.e., without filtering out any "large" marginals). This results in RAP answering approximately 50x more queries at its peak than in Aydore et al.'s initial evaluation on filtered marginals. We then use these results to analyze the role that adaptivity plays in RAP's utility. Finally, we address the question of whether filtering the large marginals out of RAP's evaluation significantly impacts its utility in order to determine if the filtering criterion is a reasonable heuristic to apply to reduce RAP's computational burden in future evaluations. This improved implementation and reevaluation, taken together, conclusively demonstrates that RAP is a feasible and valuable mechanism for practical, real-world use cases. Furthermore, in conjunction with our improved implementation, our findings enable new capabilities such as more efficient search strategies for optimal T and K parameters.

Evaluation Datasets

As in prior works on evaluating DP mechanisms that answer statistical queries [ABK+21, VTB+20, MSM19], all empirical evaluations use the ADULT [Fra10] and LOANS [VTB+20] datasets with the same preprocessing. Table 2 contains a high level description of each dataset.

3.4.1 k-way Marginal Evaluation of RAP

To begin RAP's reevaluation, we concretely establish its utility on a larger portion of the query space than previously considered by Aydore et al. Specifically, we evaluate RAP's present error for answering uniformly random workloads of 3-way marginals across a range of parameters on both the ADULT and LOANS datasets, and we do so *without*

Primary Mechanism	RAP
Baseline Mechanisms	All-0, GM
Utility Measure	err_P
D	ADULT, LOANS
ϵ	$0.01, 0.1, 1$
δ	$1/\lvert D\rvert^2$
$\lvert W\rvert$	$1, 4, 16, 64, 256$
n'	10^3
T	$1, 4, 16, 64$
K	$4, 16, 64, 256, m$
k	3

Table 3: Experimental reference table for reevaluating RAP's utility on k-way marginals.

any thresholding criterion to filter out "large" marginals. This results in RAP answering approximately 50x as many queries as in its original evaluation by Aydore et al. Table 3 provides a reference for the parameter ranges in this experiment. For each setting of parameters, we evaluate the adaptive variant of RAP across a range of T and K values and report the combinations that achieve minimal present error. We separately evalaute the non-adaptive ($T = 1, K = m$) variant of RAP across the same range of parameters in order answer the question of whether or not there is any benefit to RAP's adaptivity. Additionally, as baselines, we evaluate the present utility of the All-0 and GM mechanisms, enabling us to put the utility of RAP into context. The results of this experiment are visualized in Figure 2.

There are several immediate conclusions that can be drawn from these results. The first is that while the non-adaptive variant of RAP achieves lower error than the GM baseline, its utility is nearly identical to the All-0 baseline for all but the smallest workload sizes. This result likely stems from the fact that the answers to the large majority of a marginal's consistent queries are 0 or nearly 0, with only a small percentage of answers having larger values. Since the non-adaptive variant of RAP first privatizes the answers to all queries, in the synthetic dataset optimization procedure it is likely unable to distinguish between the few answers that are truly larger than 0 vs. the outliers that are only large due to random chance. The second conclusion is that the adaptive variant of RAP achieves significantly lower present error than the non-adaptive RAP variant as well as the baselines. This implies that RAP's adaptivity is critical for achieving low error, and thus warrants a more thorough investigation into T and K's precise impact on utility.

3.4.2 Role of Adaptivity

In this next experiment, we seek to understand the precise impact that T and K have on RAP's utility. From Figure 2, we are only able to glean that RAP typically achieves minimal error via smaller values of T in conjunction with relatively larger values of K. However, these values of T and K vary dramatically across parameter settings and datasets. Moreover, Figure 2 provides no information about RAP's utility for T and K combinations that did not achieve minimal error. To better understand the role these parameters play in RAP's utility, we examine the present error of the adaptive variant of RAP for every (T, K) pair across the same parameter settings from Table 3. The results of this experiment are shown in Figures 3 and 4.

The heatmaps in both figures provide interesting insight into RAP's adaptivity. In Figure 3, with ϵ fixed at 0.1, we see that there is no single (T, K) value or region that consistently achieves minimal error across all workload sizes. Instead, we notice that at each workload size, there is some diagonal banding at around a fixed region of $T \cdot K$ that achieves approximately minimal error. That is, for any particular workload size, let (T^*, K^*) denote the T and K value that induces minimal error for RAP across our considered range of T, K values, and let $c^* := T^* \cdot K^*$. We see that for other (T, K) pairs such that $T \cdot K \approx c^*$, the corresponding error is typically comparable to the minimal error. Moreover, we see that as $T \cdot K$ diverges from c^*, RAP's error increases essentially monotonically. We hypothesize that for $T \cdot K \ll c^*$, RAP's error is relatively high because RAP had not answered and optimized over a sufficient number of queries. For $T \cdot K \gg c^*$, we hypothesize that RAP's error is relatively high because the privacy

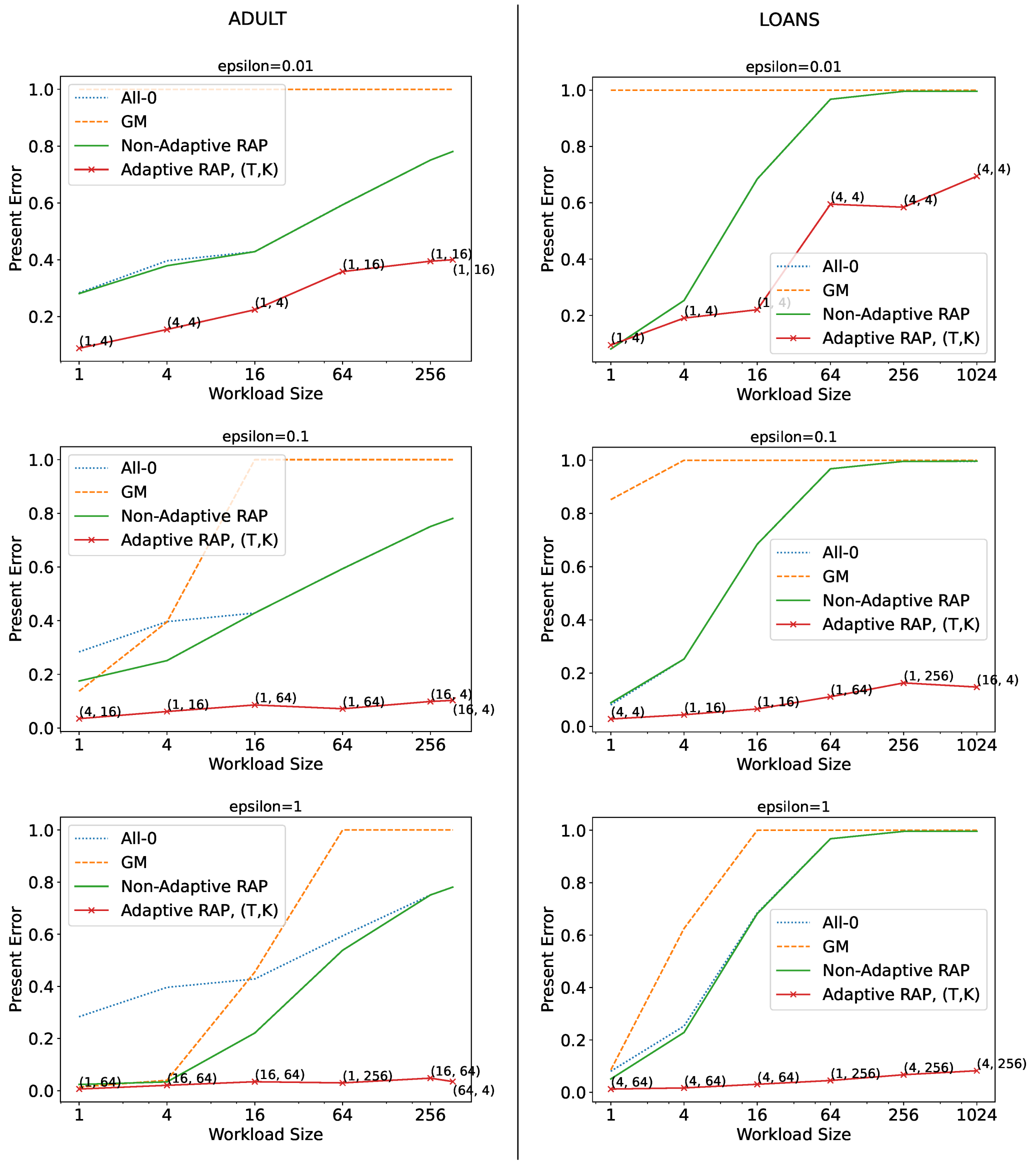

Figure 2: Present error across a range of parameters and datasets for the adaptive and non-adaptive variants of `RAP`, the `GM` baseline, and the `All-0` baseline. Present error for the adaptive variant of `RAP` is computed as the minimal error across the range of T and K values (with the specific (T, K) pair that achieved the minima reported at each point).

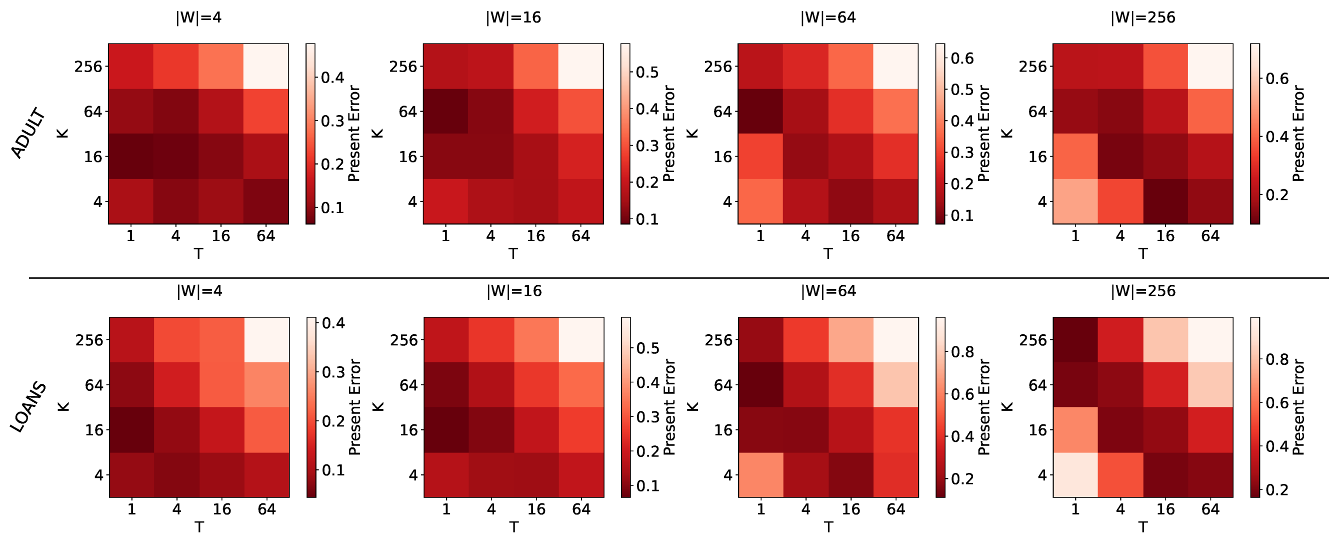

Figure 3: Present error across a range of workload sizes with $\epsilon = 0.1$ for the adaptive variant of RAP at every combination of T and K value considered.

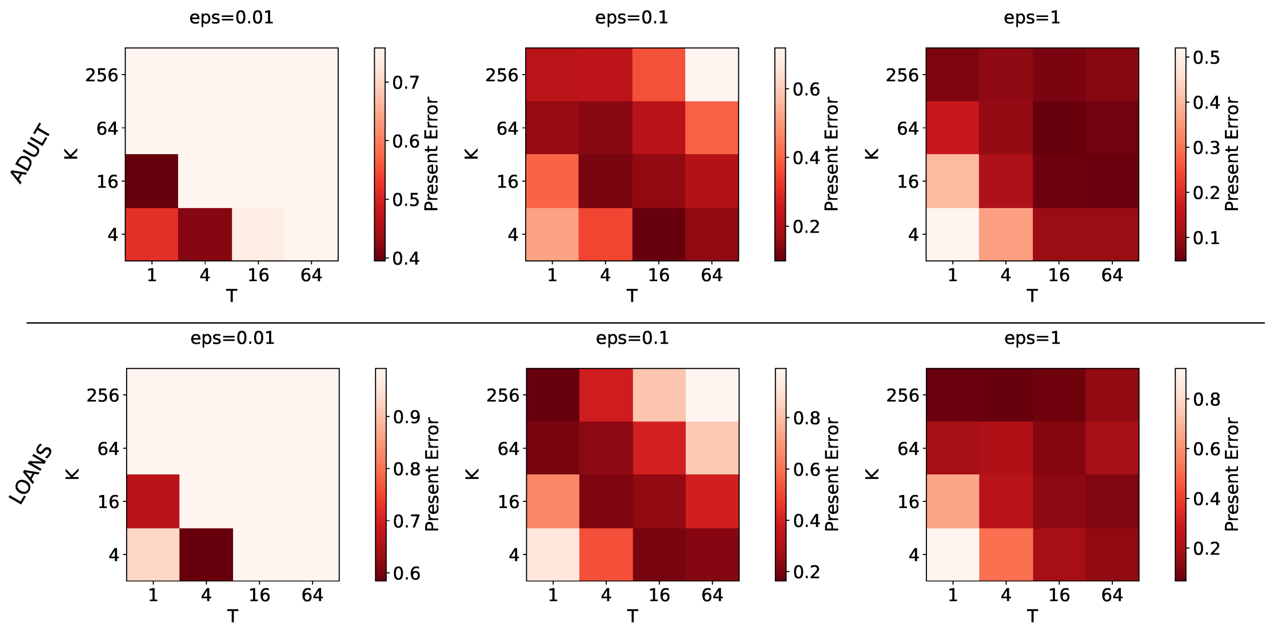

Figure 4: Present error across a range of ϵ values with $|W| = 256$ for the adaptive variant of RAP at every combination of T and K value considered.

budget is spread too thin across across answering a large number of queries, resulting in RAP utilizing overly noisy queries to optimize its underlying synthetic dataset.

These hypotheses are supported by the results in Figure 4. Specifically, as ϵ becomes larger, not only does the minimal error of RAP decrease, but the T and K values that achieve the minimal error (along with their corresponding diagonal bands) are pushed to increasingly large values. Taken together, these results imply that in order to achieve low error, RAP primarily requires answering and optimizing over a *specific number* of queries — it is less important whether those queries are answered in small batches over a large number of adaptive rounds or in large batches over a small number of adaptive rounds.

This finding is important to RAP's usefulness in practice, as it motivates improved search strategies for optimal (T, K) values. Improved search strategies (beyond the naive $N \times N$ grid search that we performed) are important for two reasons.

1. Evaluating RAP across a range of T and K values can be computationally expensive. Thus, improved search strategies would decrease the computational cost. Alternatively, at a fixed computational cost, improved search strategies would allow RAP to be evaluated across a larger set of T and K values.

2. In practice, each evaluation of RAP on any (T, K) setting consumes a portion of the privacy budget, even if only the optimal setting is chosen in the end. Thus, reducing the total number of evaluated (T, K) settings enables more efficient use of the overall privacy budget.

We provide one example of an improved search strategy over the naive $N \times N$ grid search strategy as follows. First, the observed monotonicity of present error about c^* could be leveraged to binary search for a $c := T \cdot K$ setting along the positive diagonal that achieves approximately minimal error. Then, a linear search across all (T', K') settings such that $T' \cdot K' = c$ could be performed to compute the setting that achieves minimal error. Relative to the grid search, this strategy would yield an $\mathcal{O}(N)$ factor improvement both in the portion of the privacy budget consumed as well as in the computational cost.

3.4.3 Utility Impact of Filtering Marginals

In the final experiment, we analyze what impact filtering out marginals with "too many" consistent queries has on RAP's utility. Recall that in Aydore et al.'s evaluation, as a heuristic to reduce the computational burden of experimentally evaluating RAP, any marginal was removed from consideration if it contained more consistent queries than the number of records in the underlying dataset. Here, we compare how RAP's utility is affected by this marginal filtering criterion. We initiate this comparison by reevaluating RAP with and without the filtering criterion. We do so across the range of parameters in Table 3, and we record the minimal present error of RAP at each parameter setting across all (T, K) pairs. We then perform two analyses on these results, one focusing on how the workload size affects RAP's present error with and without marginal filtering, and another analyzing how the total number of queries affects RAP's present error. We conclusively determine that RAP's present error is impacted by filtering large marginals. More specifically, we find that when holding the number of queries that RAP evaluates constant, filtering large marginals *increases* RAP's present error.

Influence of Workload Size on Utility Aydore et al. hypothesized that removing the marginal filtering criterion would cause RAP's present error to increase comparably to the error increase induced by increasing the workload size. To test this hypothesis, we perform a standard nested regression analysis [GH06] on the RAP evaluation results. For brevity, we state the steps of this analysis and then immediately jump to the results, deferring the regression details to Appendix A.

At the high level, the steps for this analysis are as follows. For the ADULT and LOANS datasets separately, we define a full regression model to account for the following three variables' (and their interactions') impact on RAP's present error: the DP level ϵ, the workload size $|W|$, and whether the marginal filtering criterion was applied. We also define a restricted regression model that accounts for ϵ and $|W|$, but does not distinguish whether or not a result had the marginal filtering criterion applied. Following the standard approach for a nested regression analysis, we first determine whether the full regression model is a good fit for the RAP evaluation results (based on the fitted model's adjusted r^2 value, F-statistic p-value, and omnibus p-value). We then compare the fit of the full model to the fit of

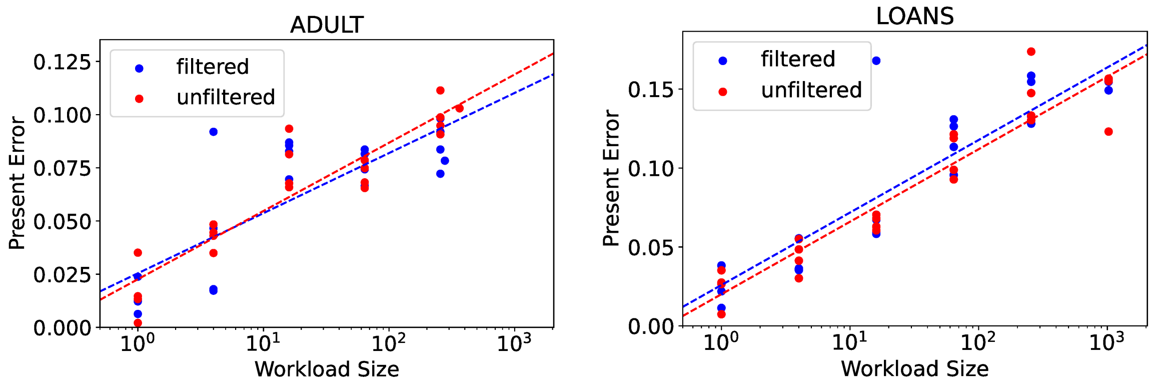

Figure 5: Regression models for each dataset of RAP's present error vs. workload size for results from filtered and unfiltered marginals, at $\epsilon = 0.1$.

the restricted model by performing a likelihood ratio test, analyzing the p-value of the resulting χ^2 statistic. Since the full model only differs from the restricted model in that it accounts for whether the marginal filtering criterion was applied, we can conclude that if the fit of the full model is both statistically sound and statistically significantly better than that of the restricted model, then the marginal filtering criterion impacts RAP's present error.

From this analysis, Figure 5 shows the fitted full regression model on both datasets with ϵ fixed at 0.1. We find that the full regression models for both datasets fit the RAP evaluation results well. Thus, we perform the aforementioned likelihood ratio test against the restricted models for each dataset. The corresponding p-values for the models on the ADULT and LOANS RAP evaluations were 0.026 and 0.623 respectively.¶ The small p-value for the model corresponding to the RAP evaluations on the ADULT dataset enables us to conclude that the marginal filtering criterion does have an impact on RAP's present error. However, the coefficients (and their corresponding p-values) in the full regression model do not indicate any clear, statistically significant trend for how the present error is impacted by the workload size when comparing the filtered vs. unfiltered RAP evaluations. Moreover, regardless of the workload size, due to the lack of significance in many of the coefficients' p-values, we are unable to use this model to confidently determine the marginal filtering criterion's impact on RAP's present error. Thus, although we are able to conclude that incorporating the marginal filtering criterion into RAP's evaluation does impact its present error, we are unable to confirm Aydore et al.'s hypothesis on the precise nature of this impact.

Influence of Number of Queries on Utility We now perform a more direct analysis of the marginal filtering criterion's impact on RAP's utility. Our previous regression analysis assessed Aydore et al.'s hypothesis regarding the filtering criterion's influence on RAP's present error as a function of workload size. However, the filtering criterion does not affect workload size directly — it only affects the total number of queries consistent with the marginals in the workload. As such, we believe that a more informative assessment would be to analyze the marginal filtering criterion's influence on RAP's present error as a function of the total number of consistent queries that it evaluates.

We perform this assessment using precisely the same statistical analysis and regression models as before, only now having the full and restricted models account for the total number of queries rather than workload size. Figure 6 shows the fitted full regression models on both datasets with ϵ fixed at 0.1. Again, the full regression models for both datasets fit the RAP evaluation results well, allowing us to then test these full models against their corresponding restricted models. The corresponding p-values of the likelihood ratio tests for the models on both the ADULT and LOANS RAP evaluations were less than 0.0001, indicating that the filtering criterion has a statistically significant impact on RAP's present error (for both datasets this time). The results from the figure for both datasets visually

¶We report the individual p-values for all statistical hypotheses tested. However, we control the family-wise error rate α (i.e., the probability α that at least one "false positive" finding will occur) using the Holm–Bonferroni method [Hol79]. At the $\alpha = 0.05$ level, no conclusions based on the individual p-values change when the Holm–Bonferroni method is applied.

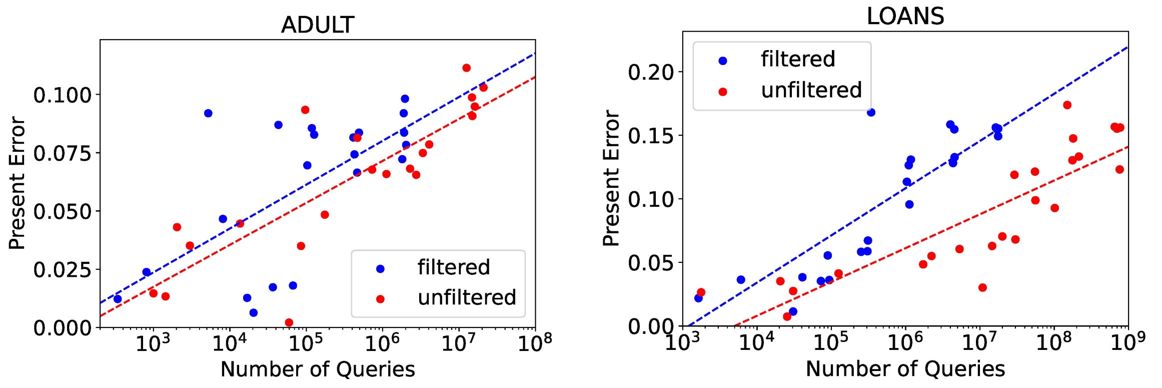

Figure 6: Regression models for each dataset of RAP's present error vs. number of queries for results from filtered and unfiltered marginals, at $\epsilon = 0.1$.

imply that including the filtering criterion increases RAP's present error for any given number of queries, and that this increase worsens as the total number of queries grows. By examining the coefficients (and their corresponding p-values) of the full regression models on both datasets, we confirm that this visual trend holds statistically as well.

These results match intuition: in order for a result with the filtering criterion to have approximately the same number of queries as a result without, the result with filtering would likely have corresponded to a larger sized workload. A larger size workload with the same number of queries implies a more diverse set of queries, whereas a smaller workload with the same number of queries implies a less diverse set of queries with sparser support (i.e., more of the queries evaluate to 0). Thus, we conclude that Aydore et al.'s initial evaluation of RAP — especially for the highly filtered 5-way marginals — likely overestimates RAP's present error. Moreover, this finding motivates a new branch of work on large-scale query answering for the separate cases of when the queries have dense support vs. sparse.

4 Extending RAP's Applicability

In this section, we address our third contribution for the setting where queries are prespecified: extending RAP's applicability by expanding the class of queries that it is able to evaluate. We begin by discussing the motivation behind this contribution. We then describe what we expand the query class to (r-of-k thresholds) and how we accomplish it. Finally, we detail the empirical evaluations we perform on RAP within this expanded query class to quantify its utility and feasibility, finding that RAP efficiently evaluates r-of-k thresholds with high utility.

4.1 Motivation

We contextualize the motivation for this contribution by considering the contributions of prior works. Prior work on answering statistical queries in practical settings has been focused on relatively simple classes of statistical queries — most popularly, k-way marginals (Definition 2.2), as these are a useful query class which is evaluable within a reasonable computational budget [BCD+07, TUV12, GHRU13, CTUW14]. Aydore et al.'s claim is that their gradient-based RAP mechanism [ABK+21] is able to answer queries from richer classes. In addition to evaluating k-way marginals, they demonstrated this claim by briefly evaluating a new class of queries, 1-of-k thresholds (Definition 2.3). However, 1-of-k thresholds are essentially a negation of k-way marginals. As such, Aydore et al. were able to evaluate RAP on 1-of-k thresholds by reusing virtually the same class of EEDQs and the same underlying implementation as they used for k-way marginals. Thus, although their evaluation demonstrated that RAP attains high utility on both query classes, these choices of query classes were not fully convincing in demonstrating that RAP is effective for answering truly richer classes of queries. Therefore, it remained an open question whether RAP is able to answer richer, more general query classes.

4.2 Expanding the Query Class

To extend RAP's applicability, we develop the mathematical and computational machinery necessary for RAP to evaluate a class of queries which generalizes both k-way marginals and 1-of-k thresholds: r-of-k thresholds (Definition 2.4). We first describe this query class in detail, then derive its corresponding EEDQs. Finally, we show how we optimize the derived EEDQs to be more efficiently evaluable, greatly reducing RAP's per-query evaluation time.

4.2.1 Generalizing to r-of-k Thresholds

Informally, an r-of-k threshold query counts what fraction of datapoints in the dataset have at least r out of the k specified attributes. Thus, it strictly generalizes both k-way marginals (when $r = k$) and 1-of-k thresholds (when $r = 1$). r-of-k thresholds are a useful generalization because they allow for more expressive, dynamic queries beyond the rigid "everything" ($r = k$) or "anything" ($r = 1$) queries that were previously studied.

The challenge when expanding RAP's evaluation to r-of-k thresholds is deriving corresponding EEDQs. r-of-k thresholds cannot trivially reuse the EEDQs relied upon by Aydore et al. to evaluate k-way marginals and 1-of-k thresholds. Thus, we must derive new EEDQs for r-of-k thresholds, and we accomplish this by generalizing the EEDQs of k-way marginals and 1-of-k thresholds. Towards this, we first reframe the standard definition of r-of-k thresholds to enable explicit accounting of all possible combinations of matching and non-matching terms.

Definition 4.1 (r-of-k thresholds, Alternative)**.** An r-of-k threshold query $q_{\phi_{S,y,r}}$ is a statistical query whose predicate is specified by a positive integer $r \leq k$, a set S of k features $f_1 \neq \ldots \neq f_k \in [d]$, and a target $y \in (\mathcal{X}_{f_1} \times \cdots \times \mathcal{X}_{f_k})$. Let $\mathcal{R}$ denote the set of all partitions (R_+, R_-) of the k features in S, such that each $|R_+| \geq r$ and each corresponding $R_- = S - R_+$. The predicate $\phi_{S,y,r}$ is then given by

$$\phi_{S,y,r}(x) = \begin{cases} 1 & \text{if } \bigvee_{(R_+,R_-)\in\mathcal{R}} \left(\bigwedge_{i\in R_+} (x_{f_i} = y_i) \bigwedge_{i\in R_-} (x_{f_i} \neq y_i) \right) \\ 0 & \text{otherwise.} \end{cases}$$

Note that at most one partition in $\mathcal{R}$ will satisfy the predicate.

We now use this equivalent definition of r-of-k thresholds queries to design corresponding EEDQs. For k-way marginals, Aydore et al. used *product queries* (Definition 2.7) as EEDQs, which simply compute the product of a datapoint's values at the k specified indices. For r-of-k threshold queries, we generalize product queries in the following ways. First, we expand the product queries to explicitly include both positive and negated terms, which we refer to as *generalized product queries.*

Definition 4.2 (Generalized Product Query)**.** Given two disjoint subsets of features $T_+, T_- \subseteq [d']$, the generalized product query $\hat{q}_{\hat{\phi}_{T_+,T_-}}$ is a surrogate query parameterized by $\hat{\phi}_{T_+,T_-}$ which is defined as

$$\hat{\phi}_{T_+,T_-}(x) = \prod_{i\in T_+} x_i \prod_{i\in T_-} (1 - x_i).$$

Informally, a generalized product query effectively serves as a "sub"-EEDQ for the conjunction portion of a single partition of $\phi_{S,y,r}(x)$ in Definition 4.1.

Then, leveraging this alternative definition of r-of-k thresholds together with generalized product queries, we define a new class of EEDQs in Definition 4.3: *polynomial threshold queries.*

Definition 4.3 (Polynomial Threshold Query)**.** Given a subset of features $T \subseteq [d']$ and integer r, let Υ denote the set of all partitions (T_+, T_-) of T such that each $|T_+| \geq r$ and each corresponding $T_- = T - T_+$. The *polynomial threshold query* $\hat{q}_{\hat{\phi}_{T,r}}$ is a surrogate query parameterized by $\hat{\phi}_{T,r}$ which is defined in terms of the generalized product query predicates as

$$\hat{\phi}_{T,r}(x) = \sum_{(T_+,T_-)\in\Upsilon} \hat{\phi}_{T_+,T_-}(x).$$

Informally, a polynomial threshold query computes the sum of generalized product queries across all $\sum_{t=r}^{k} \binom{k}{t}$ partitions of T, where T is constructed identically as in Lemma 2.8; i.e., for every $i \in S$, we include in T the coordinate corresponding to $y_i \in \mathcal{X}_{f_i}$.

4.2.2 Optimizing the Evaluation of Polynomial Threshold Queries

Evaluating polynomial threshold queries can be computationally expensive due to their combinatorial expansion and summation of generalized product query predicates. Therefore, optimizing their definition to be efficiently evaluable is of utmost importance for enabling `RAP` to evaluate large sets of r-of-k thresholds. Towards this, we present two optimizations that can be used together, which significantly improve the practical runtime of `RAP`.

The first optimization is inspired by Aydore et al.'s implicit reduction of 1-of-k threshold queries to k-way marginal queries. They accomplished this by recognizing that a 1-of-k threshold predicate is the negation of a k-way marginal predicate on a negated datapoint; i.e., $\phi_{S,y,1}(x) = 1 - \phi_{S,y,k}(1-x)$. This equivalence enabled them to efficiently reuse the k-way marginals' EEDQs (product queries) in `RAP`'s evaluation. Applying this concept more generally to computing an r-of-k threshold predicate $\phi_{S,y,r}(x)$, the idea is that when $r \leq k/2$, it is logically equivalent to compute the negation of a corresponding predicate (with $r' = k - r + 1$) on the negated datapoint; i.e., $\phi_{S,y,r}(x) = 1 - \phi_{S,y,r'}(1-x)$. The benefit of utilizing this equivalence when using a polynomial threshold query as the EEDQ to evaluate $\phi_{S,y,r}(x)$ is that *at most* $\lceil k/2 \rceil$ different partition sizes now need to be computed over, compared to at most k when not utilizing this equivalence. The computational savings from utilizing the equivalence are especially apparent when r is small, as it leads to an exponential (in k) reduction in the required number of predicate evaluations.

For the second optimization, the goal is to eliminate the need to explicitly account for the negated terms in our alternative definition of r-of-k thresholds (Definition 4.1), as this in turn necessitates the computation of the product of negated values in generalized product queries (Definition 4.2). Removing the conjunction over negated terms from Definition 4.1 yields a logically equivalent predicate; i.e.,

$$\phi_{S,y,r}(x) = \begin{cases} 1 & \text{if } \bigvee_{(R_+,R_-)\in\mathcal{R}} \bigwedge_{i\in R_+} (x_{f_i} = y_i) \\ 0 & \text{otherwise.} \end{cases}$$

However, more than one partition of $\mathcal{R}$ may now satisfy the predicate. As a result, analogously eliminating the product of negated values from the generalized product query definition (reducing it to a standard product query) would cause the summation in the polynomial threshold query's definition (Def 4.3) to overcount. To eliminate computing the product of negated values while simultaneously remedying this overcount, we utilize the principle of inclusion-exclusion to equivalently redefine polynomial threshold queries purely in terms of standard product queries (Definition 2.7).

Definition 4.4 (Polynomial Threshold Query, Inclusion-Exclusion)**.** Given a subset of features $T \subseteq [d']$ and integer r, let $\Upsilon(i)$ denote the set of all i-size combinations of features in T for $i = r \ldots k$; i.e., each $T_i \in \Upsilon(i)$ is such that $|T_i| = i$ and $T_i \subseteq T$. The polynomial threshold query $\hat{q}_{\hat{\phi}_{T,r}}$ parameterized by $\hat{\phi}_{T,r}$ can be defined in terms of product query predicates $\hat{\phi}_{T_\cdot}$ as

$$\hat{\phi}_{T,r}(x) = \sum_{i=r}^{k} (-1)^{i-r} \binom{i-1}{i-r} \sum_{T_i \in \Upsilon(i)} \hat{\phi}_{T_i}(x).$$

Utilizing this redefinition of polynomial threshold queries reduces the number of arithmetic operations by nearly half relative to the original definition (when $r > k/2$, which we assume without loss of generality by simultaneously utilizing the first optimization in this section). In our subsequent experiments with r-of-4 thresholds (Section 4.3), this reduction in operations results in a maximal runtime improvement of approximately 40% for evaluating the polynomial threshold queries.

4.3 Evaluating `RAP` on r-of-k Thresholds

With the class of EEDQs derived, the only question that remains is how well `RAP` is able to utilize the EEDQs to answer prespecified sets of r-of-k thresholds. We investigate this question by evaluating how the various inputs to `RAP` affect its present utility and runtime.

Primary Mechanism	`RAP`
Baseline Mechanisms	`All-0`, `GM`
Utility Measure	err_P
D	ADULT, LOANS
ϵ	$0.1, 1$
δ	$1/\lvert D\rvert^2$
$\lvert W\rvert$	$1, 4, 16, 64, 256$
n'	$500, 1000, 2000$
T	$1, 4, 16, 64$
K	$4, 16, 64, 256$
r	$1, 2, 3, 4$
k	4

Table 4: Experimental reference table for evaluating r-of-k thresholds with `RAP`.

4.3.1 Utility on r-of-k Thresholds

To begin, we evaluate the present utility of `RAP` on r-of-k thresholds, with k fixed at 4. As in our prior experiments in Section 3, we contextualize `RAP`'s utility by comparing against the utilities of the `All-0` and `GM` baseline mechanisms. We then evaluate the utility of each mechanisms across a range of r values, ϵ values, datasets D, workload sizes $|W|$, and synthetic dataset sizes n', and across the same T, K values for `RAP` as before. Table 4 contains a summary of the precise parameter values.

Figure 7 displays the results of this experiment for $n' = 1000$, showing the minimal present error of `RAP` across all T, K values considered alongside the present error of the baseline mechanisms. The present error of both baseline mechanisms are as expected, with the `All-0` mechanism's present error having a clear and straightforward dependence on r, whereas the `GM` mechanism's present error is independent of r. Immediately, we see that `RAP` significantly outperforms the baseline mechanisms in all settings. Across the r values, we find that `RAP` achieves its minimal present error at $r = 4$ (i.e., 4-way marginals). Although `RAP`'s present error for $r < 4$ is not much greater than for $r = 4$, we find no further obvious relationship between `RAP`'s present error and r.

To understand the role of that `RAP`'s adaptivity plays in this experiment, in Figure 8 we visualize `RAP`'s present error for each individual combination of T, K settings considered. Just as with 3-way marginals in Section 3.4.2, we find that the same adaptivity behavior emerges with 4-way marginals ($r = 4$); i.e., `RAP` primarily needs to evaluate a specific number of queries to achieve low present error, regardless of whether those queries are evaluated jointly in a small number of adaptive rounds or individually across a large number of adaptive rounds. However, we find that this behavior no longer holds for $r < 4$. Instead, the only consistent pattern that we find for $r < 4$ in this figure (which holds across other workload sizes and ϵ values as well) is that `RAP` achieves its minimal present error when the number of adaptive rounds is relatively large but the number of selected queries per round of adaptivity is relatively small. Since executing `RAP` for a large number of adaptive rounds is computationally expensive, this finding motivates future work on reducing the necessary number of rounds of adaptivity. This could potentially be done by more strategically selecting the set of queries in each round — for instance, by considering their expected joint impact on `RAP`'s present error in the next optimization step, rather than selecting the individual queries with highest present error independently.

4.3.2 Effect of Synthetic Dataset Size

Lastly, we investigate how `RAP`'s synthetic dataset size n' affects its present error and runtime. Conceptually, n' controls `RAP`'s learning capacity — the larger n', the better the answers to the queries should be. However, since optimizing large synthetic datasets is computationally expensive, n' cannot be taken arbitrarily large. Similarly, when the synthetic dataset size is too small, the optimization problem becomes underparameterized, which also results in a computationally expensive optimization process. Aydore et al. empirically confirmed this utility–computation trade-off for `RAP` with k-way marginals, where they found that setting $n' = 1000$ served as a good balance between utility

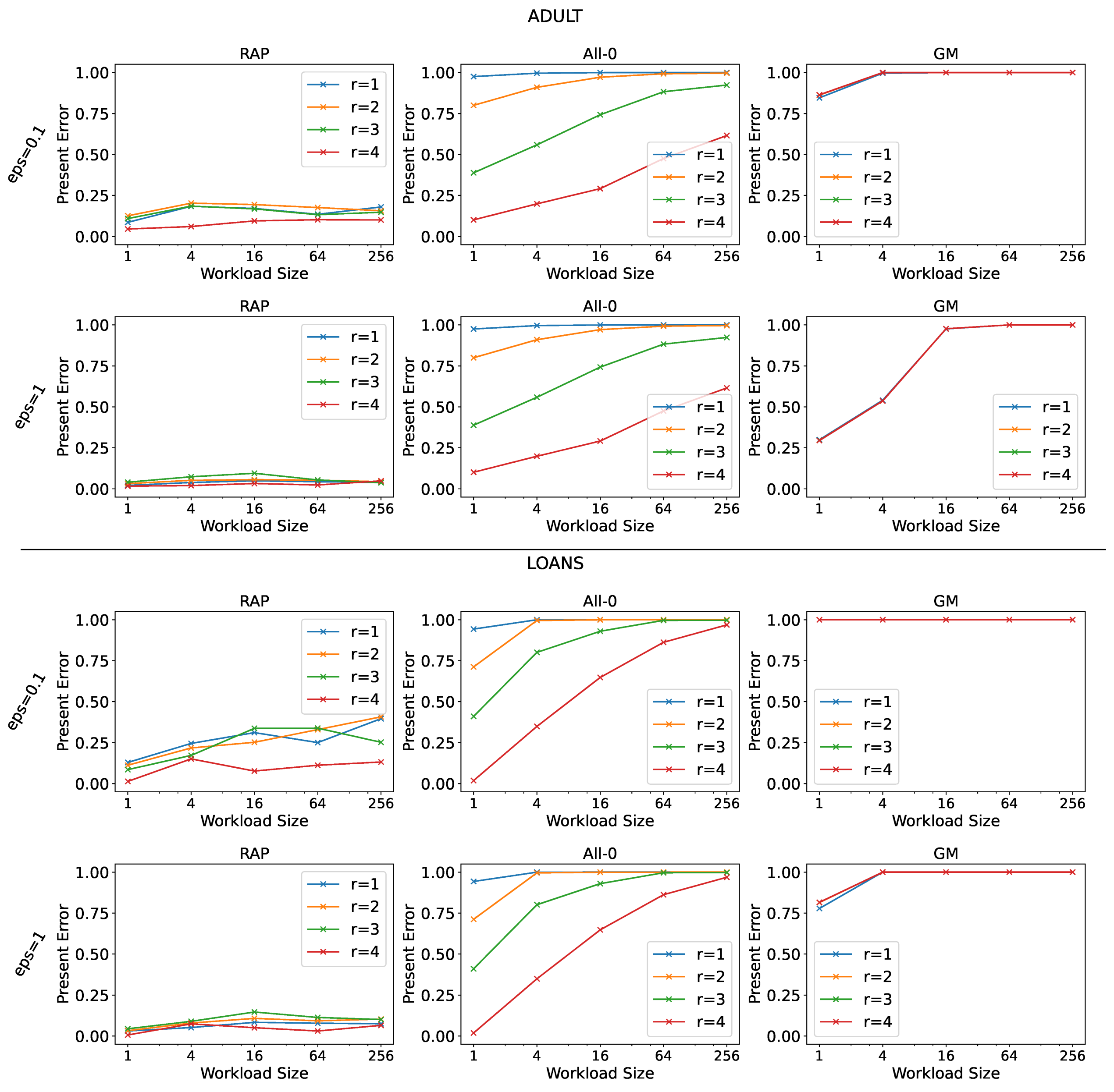

Figure 7: RAP's minimal present error across all T, K values considered alongside present error of the baseline mechanisms.

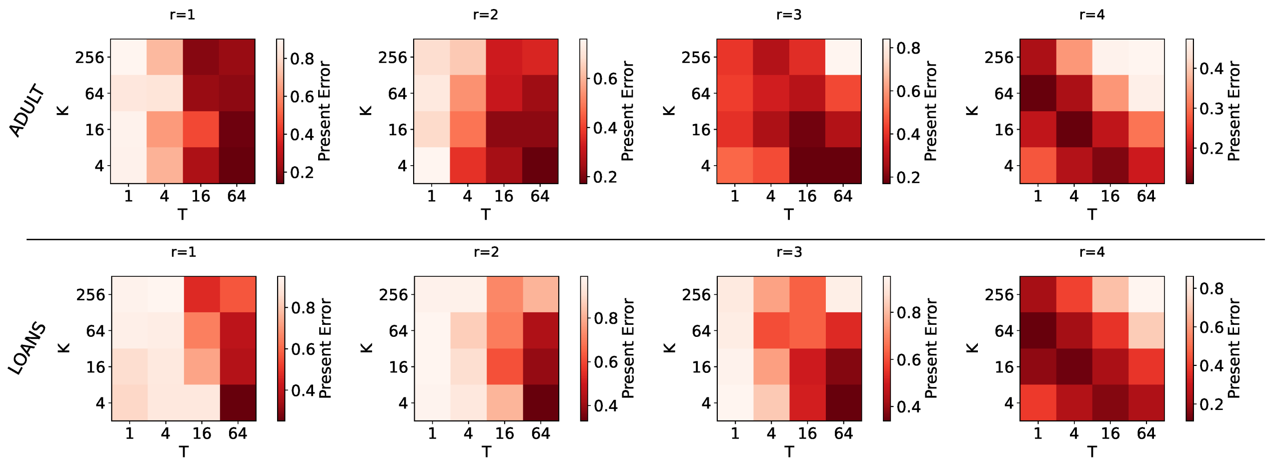

Figure 8: RAP's present error at each T, K value considered on a workload of 64 r-of-k thresholds with $\epsilon = 0.1$.

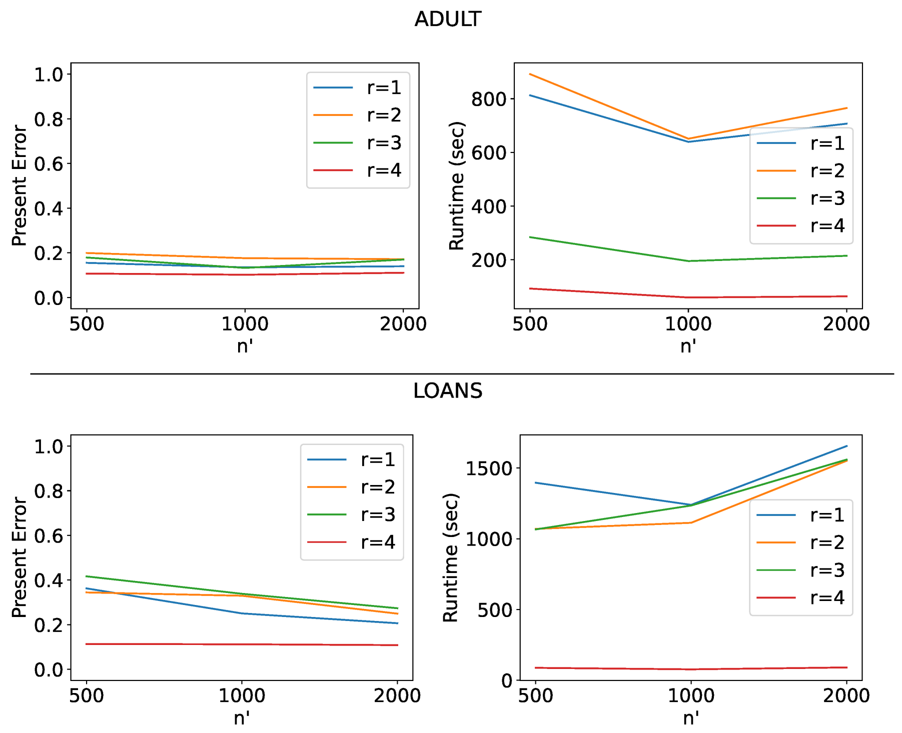

Figure 9: RAP's present error and runtime as a function of the synthetic dataset size on a workload of 64 r-of-k thresholds with $\epsilon = 0.1$.

and runtime for (filtered) 3-way and 5-way marginals.

We evaluate this trade-off on (unfiltered) r-of-4 thresholds, with the results shown in Figure 9. For each setting of r, we find that increasing n' generally results in a mild reduction of RAP's present error, but that at $n' = 1000$ RAP often attains minimal or near-minimal runtime. This mirrors Aydore et al.'s results, and thus supports their findings regarding RAP's utility–computation trade-off. However, one interesting new finding is the effect that r has on RAP's runtime. Apriori, we expected that RAP would have the shortest runtime when evaluating r-of-4 thresholds with $r \in \{1, 4\}$, and that their runtimes would be comparable. This is because at $r \in \{1, 4\}$, RAP has the least arithmetic operations to perform in order to evaluate each predicate (compared to $r \in \{2, 3\}$, refer to Section 4.2.2 for details on predicate evaluation). Although we confirm that RAP acheives minimal runtime at $r = 4$, we find that nearly the opposite holds true for $r = 1$, which induces up to a 20x longer runtime. This increase in runtime is primarily explained by our prior observation that for $r < 4$, RAP achieves its maximal utility via a larger number of adaptive rounds (where RAP's runtime appoximately linearly increases with the number of rounds).

However, even with this jump in runtime taken into consideration, we find that RAP is a highly performant mechanism for evaluating large sets of queries. For instance, consider the worst-case runtime at $n' = 1000$ in Figure 9, which occurs where RAP answered a workload of 64 1-of-4 thresholds on the LOANS dataset. Here, RAP answered approximately 3.5×10^7 individual consistent queries in 1,240 seconds — a rate of over 28,000 queries per second. Based on these findings, we conclude that RAP is highly efficient for answering large sets of r-of-k thresholds.

5 Understanding RAP's Generalizability

In this final section, we propose a new and realistic intermediate setting that lies between the classic settings of having full knowledge of all queries in advance (i.e., the prespecified queries setting) vs. having no knowledge of which queries will be posed. We begin by concretely defining this new partial knowledge setting along with a generalization-based measure of utility for mechanisms operating within it. We then address our final contribution by empirically evaluating RAP's utility to determine its suitability in the new setting.

Motivation

In statistical modeling, and especially in the subfield of synthetic data generation, the primary goal is not to generate a model or a synthetic dataset that answers a prespecified set of queries well. Rather, the goal is to generate a model or synthetic dataset that *generalizes* well to future queries [Vap99, MRT12]. When it comes to differentially private mechanisms for answering statistical queries through a synthetic dataset, prior utility analyses have focused on either: (a) how well those mechanisms answer the prespecified set of queries, or (b) theoretically bounding how well the mechanisms can answer any class of queries in the worst-case. For example, the utility of RAP (and the related practical mechanisms which preceded it) had previously been based on solely the answers to the prespecified workload; e.g., present utility. Experimentally evaluating a mechanism's present utility is straightforward: simply report the error of the highest error query from the prespecified query set. However, in some settings, it may be more useful to understand how well the mechanism can answer future queries. Towards this, theoretical bounds can provide strong guarantees for the mechanism's worst-case utility across an entire query class [BLR08, DNR+09, DRV10, HLM12, TUV12]. The drawback to using these theoretical bounds in practical settings is that they may be overly pessimistic, especially if the queries posed in the future are highly similar to the queries that were used to generate the synthetic dataset. This apparent disparity between the utility suggested by theoretical analyses and the actual utility that may be observed in practice is nearly identical to the disparity that famously exists between utility analyses in theoretical vs. empirical machine learning research [Vap98, BM06, SSBD14, NTS14, ZBH+21]. However, for answering statistical queries with DP, the theoretical worst-case bounds are currently the best tool available without introducing additional information or assumptions.

5.1 Defining the Partial Knowledge Setting

We now motivate the design of this particular partial knowledge setting, then formally define it.

Much like in the machine learning research literature, we motivate a new partial knowledge setting for the context of differential privacy based on the rationale that in some realistic settings, future queries may be similar to queries posed in the past; i.e., historical queries. For instance, the U.S. Census Bureau periodically collects sensitive data for the decennial census, and routinely allows researchers to securely pose queries directly on the collected data. Because similar data is being collected each decennial census, it is very likely that some of the queries analysts pose on one census dataset will be similar to the queries that analysts pose on the next census dataset.

We formalize this intuition on partial query repeatability for r-of-k thresholds in a general manner in Definition 5.1. For ease of exposition, we first introduce the following notation. Let $\mathcal{T}$ be an arbitrary distribution over thresholds, and let $Q \leftarrow \mathcal{T}$ denote the vector of all consistent queries Q of a threshold randomly drawn from distribution $\mathcal{T}$. Similarly, we let $Q \xleftarrow{|W|} \mathcal{T}$ denote the vector of all consistent queries Q from a $|W|$ size workload of thresholds sampled i.i.d. from $\mathcal{T}$.

Definition 5.1 (Partial Knowledge Setting, General)**.** Let $\mathcal{T}_H$ and $\mathcal{T}_F$ be arbitrarily related distributions over thresholds. In this setting, DP mechanisms are expected to answer arbitrary future thresholds drawn i.i.d. from $\mathcal{T}_F$. However, the DP mechanisms are not provided $\mathcal{T}_F$ explicitly. Instead, DP mechanisms are provided access to partial knowledge of $\mathcal{T}_F$ via a workload W_H of "historical" thresholds sampled i.i.d. from $\mathcal{T}_H$; i.e., the mechanisms are given access to $Q_H \xleftarrow{|W_H|} \mathcal{T}_H$.

Intuitively, in this partial knowledge setting, mechanisms can utilize Q_H to learn about the underlying threshold distribution $\mathcal{T}_H$, and if $\mathcal{T}_H$ is similar to $\mathcal{T}_F$, this will, in turn, inform what areas of the threshold space future thresholds are most likely to be sampled from. The role of Q_H in this setting is analogous to the role that training data plays in machine learning; i.e., it is the concrete sample of data provided to the mechanism that the mechanism can use to attempt to generalize.

In order for the historical queries Q_H to convey useful information about $\mathcal{T}_F$ to the DP mechanism, $\mathcal{T}_H$ and $\mathcal{T}_F$ should be related. Towards this, in Definition 5.2 we specify two concrete instantiations of the partial knowledge setting which make the relationship between $\mathcal{T}_H$ and $\mathcal{T}_F$ explicit.

1. Informally, the first concrete instantiation is the *exact* partial knowledge setting, where historical thresholds are drawn from the same distribution as the future thresholds.

2. The second concrete instantiation is the *drifting* partial knowledge setting, which extends the exact partial knowledge setting. The drifting partial knowledge setting is inspired by the practical consideration that even if the historical and future thresholds distributions are initially the same, they may gradually drift apart over time.

In both settings, we ground the historical and future thresholds distributions in the observation that in practice, certain features (or combinations of features) are likely to be more relevant to analysts than other features. For instance, in the ADULT dataset, "Age" and "Years of education" might be more relevant and useful for analyses than "Capital loss amount" and "Relationship status". We model this relevance as a historical probability distribution $\mathcal{F}_H$ over the *features*, such that the probability mass corresponding to any r-of-k threshold in $\mathcal{T}_H$ corresponds to the (normalized) product of the k features' probabilities; i.e., $\mathcal{T}_H$ is the sampling distribution of k features from $\mathcal{F}_H$ without replacement. Our definition of the drifting partial knowledge setting specifically attempts to capture the practical phenomenon that if (for instance) analysts' interests are concentrated primarily in a small subset of features, then even if their interests drift over time, the analysts' new interests may still be concentrated in a small subset of different features. Based on this, we now formally define both concrete instantiations of the partial knowledge setting.

Definition 5.2 (Partial Knowledge Setting, Exact & Drifting)**.** Let $\mathcal{F}_H$ be an arbitrary historical distribution over features with $\mathcal{T}_H$ as its corresponding historical thresholds distribution. Without loss of generality, assume the features are sorted in descending order of their probability masses under $\mathcal{F}_H$; i.e., for each feature f_i with probability p_i, we have that $p_i \geq p_{i+1}$. Let $\gamma \in [0, 1]$ be a drift parameter, which defines the distributional similarity of the future distribution over features $\mathcal{F}_F$ (and correspondingly the future thresholds distribution $\mathcal{T}_F$) as follows. For

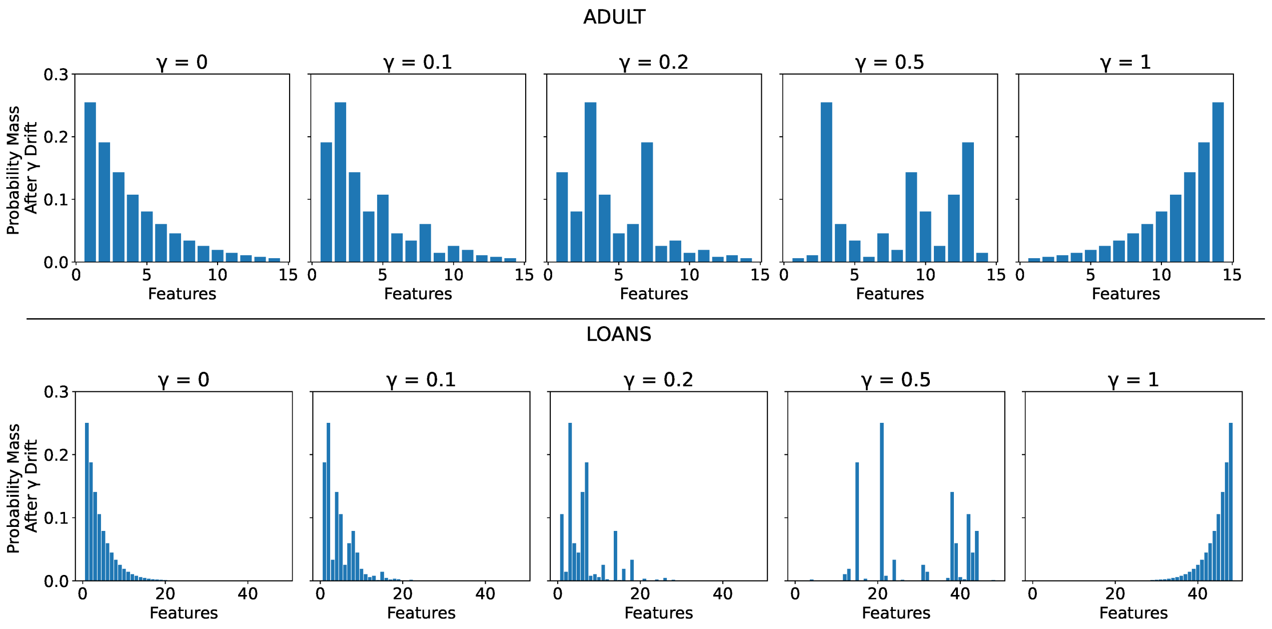

Figure 10: Examples of drifted feature distributions $\mathcal{F}_F$ across a range of drift parameters γ, with an initial Geometric distribution for $\mathcal{F}_H$ on the ADULT and LOANS datasets. Categorical features are numbered (rather than named) along the x-axis.

each probability p_i, associate the corresponding key

$$k_i = \underbrace{(1-2\gamma)}_{\text{ordering weighting}} \cdot \underbrace{\frac{d-i}{d-1}}_{\text{relative order, normalized}} + \underbrace{(1-|1-2\gamma|)}_{\text{shuffling weighting}} \cdot \underbrace{u_i}_{\text{random shuffling amount}},$$

where $u_i \overset{\text{iid}}{\sim} \text{Uniform}[0,1]$. The feature distribution $\mathcal{F}_F$ is defined by leaving the features fixed in their original ordering, but reordering the probability masses in descending order of their keys. This results in a distribution of the same concentration, but with probability masses re-assigned to potentially different features. The future thresholds distribution $\mathcal{T}_F$ is therefore the sampling distribution of k features without replacement from $\mathcal{F}_F$. When $\gamma = 0$, this procedure yields $\mathcal{T}_F = \mathcal{T}_H$, and we refer to this as the *exact* partial knowledge setting. When $\gamma > 0$, we refer to this as the *drifting* partial knowledge setting.

This model of drift is designed to maintain the concentration of the initial feature distribution $\mathcal{F}_H$ while interpolating between the exact partial knowledge setting ($\gamma = 0$) and a uniformly random reshuffling of the features' probabilities ($\gamma = 1/2$). For $0 < \gamma < 1/2$, this model induces a weighted amount of random reshuffling of probabilities in conjunction with simultaneously encouraging features' probabilities to remain "similar" to what they initially were; e.g., features with large probability masses under $\mathcal{F}_H$ are likely to retain large probability masses under $\mathcal{F}_F$. On the other end of the spectrum is the $\gamma > 1/2$ setting, where the relative orderings of probabilities become more likely to be reversed; e.g., features with large probability masses under $\mathcal{F}_H$ are likely to be assigned small probability masses under $\mathcal{F}_F$. At the extreme of this setting is $\gamma = 1$, which induces $\mathcal{F}_F$ of maximal total variation distance to $\mathcal{F}_H$ by deterministically reversing the relative ordering of the features' probabilities. Figures 10 and 11 concretely illustrate how the drift amount γ affects the distribution of future features.

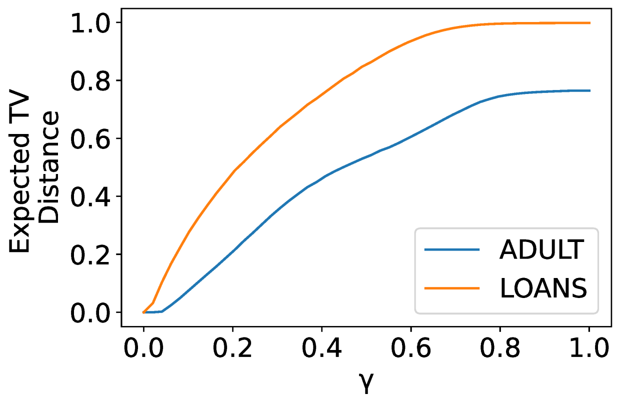

Figure 11: Effect of drift parameter γ on the total variation distance between the historical features distribution $\mathcal{F}_H$ and the future features distribution $\mathcal{F}_F$, with an initial Geometric distribution for $\mathcal{F}_H$ on the ADULT and LOANS datasets.

5.2 Measuring and Computing Utility

Having concretely defined the partial knowledge setting, we formally define a utility measure to quantify how well a mechanism can answer future thresholds based on the historical thresholds it was given access to; i.e., a measure quantifying how well the mechanism generalizes. We then describe how to empirically evaluate this defined utility measure in an efficient way.

In this setting, we are interested in the mechanism's error across its answers to the consistent queries of r-of-k thresholds drawn from $\mathcal{T}_F$. This new utility measure is based on the classic utility measure used in the prespecified queries setting (Definition 3.1), with the only difference being that the randomness of the future thresholds distribution $\mathcal{T}_F$ is now explicitly taken into account. We thus define *future utility* which we measure in terms of the negative of *future error*; i.e., a mechanism with low future error has high future utility, and vice versa. Specifically, future error is the expected absolute error taken over the randomness of both M and $\mathcal{T}_F$, formally defined as follows.

Definition 5.3 (Future error)**.** Let $a = Q(D)$ be the true answers to all queries in Q on D, and let $\tilde{a}$ be mechanism M's corresponding answers. Then err_F is the future error of mechanism M, defined as $\mathrm{err}_F(M, D, \mathcal{T}_F) = \mathbb{E}_{M(D), Q \leftarrow \mathcal{T}_F} \|a - \tilde{a}\|_\infty$, where the expectation is over the randomness of both the mechanism and future threshold distribution.

Theoretically evaluating err_F of a mechanism on *a priori* unknown threshold distributions without resorting to worst-case bounds is a challenging problem. Experimentally, however, we are able to efficiently and accurately estimate err_F for the RAP mechanism as follows:

1. Construct feature distributions $\mathcal{F}_H$ and $\mathcal{F}_F$ according to real-world phenomena, which in turn define threshold distributions $\mathcal{T}_H$ and $\mathcal{T}_F$.

2. Generate a workload W_H of historical thresholds, yielding query vector $Q_H \xleftarrow{W_H} \mathcal{T}_H$. Independently, generate a workload W_F of future thresholds, yielding query vector $Q_F \xleftarrow{W_F} \mathcal{T}_F$.

3. Provide Q_H as the input queries to RAP in order to generate a synthetic dataset.

4. Use the synthetic dataset to answer Q_F, recording the mean error (and optionally, the corresponding confidence intervals to quantify how faithfully err_F was approximated).

This evaluation approach is analogous to standard practice in empirical machine learning research where data is split into "training" and "test" sets randomly (to ensure distributional similarity) [HTFF09]. The model is then learned on the training set, and subsequently evaluated on the test set to measure how well it generalizes.

5.3 Evaluating `RAP`'s Future Utility

As our final contribution, we empirically evaluate `RAP`'s future utility for answering r-of-k thresholds. The experiments that we perform on `RAP` to understand its suitability in this new partial knowledge setting are as follows:

- Evaluating the effects that the threshold distribution concentration and the historical threshold workload size $|W_H|$ have on `RAP`'s future utility.
- Evaluating the effect that "overfitting" in the synthetic data optimization step has on `RAP`'s future utility.
- Evaluating the effect that the distribution drift amount γ has on `RAP`'s future utility.

These experiments are designed to assess the distinct new ways (beyond those in the previous prespecified queries setting) in which `RAP`'s inputs may influence its future utility.

5.3.1 Effect of Threshold Distribution Concentration & Historical Workload Size

To empirically evaluate `RAP`'s future utility in the exact partial knowledge setting, we must specify the particular threshold distribution $\mathcal{T}_H = \mathcal{T}_F$ from which we generate both the input queries Q_H and future queries Q_F used to evaluate err_F. As previously discussed, we do so by specifying feature distributions $\mathcal{F}_H$ and $\mathcal{F}_F$ that, in turn, define the threshold distributions. As a baseline, we choose what is intuitively the most challenging extreme: setting $\mathcal{F}_H$ and $\mathcal{F}_F$ to be the Uniform distribution. We expect the future utility of this baseline to be the lowest among all possible distributions since it is the least concentrated, implying that it provides the least amount of information possible to the mechanism about any particular region of the threshold space.

In an effort to model the real-world phenomena that certain features are likely to be more relevant to analysts than other features, we utilize the following two feature distributions. For a highly concentrated distribution, we use the exponentially-tailed Geometric distribution. For a mildly concentrated distribution, we use the heavy-tailed Zipfian distribution. Both distributions are commonly used in practice when modeling real-world phenomena; e.g., [MC89, YCLZ04, ZDCW12, OVL18]. We hypothesize that the highly concentrated Geometric distribution will induce high-utility results, since many of the same features in Q_H will also appear in Q_F. Analogously, we hypothesize that the mildly concentrated Zipfian distribution will induce lower-utility results (although still higher than the Uniform distribution baseline).

With a fixed threshold distribution $\mathcal{T}_H$ defined by the feature distribution $\mathcal{F}_H$, we must specify how many thresholds will be randomly sampled to form the historical threshold workload W_H (and corresponding vector of all consistent queries Q_H) that `RAP` takes as input. Obtaining a clear understanding what impact the historical workload size $|W_H|$ has on `RAP`'s future utility is important because there may be a tension between the number of historical r-of-k thresholds and `RAP`'s future utility. On the one hand, the more sampled thresholds there are, the more information `RAP` has about the underlying distribution $\mathcal{T}_F$ from which future thresholds will be generated. This suggests that the more historical r-of-k thresholds there are, the higher `RAP`'s future utility should be. On the other hand, to optimize `RAP`'s underlying synthetic dataset, its privacy budget is split between all queries consistent with the historical thresholds. This implies that the more historical thresholds there are, the more noise will be added to each consistent query's answer, which seems to suggest that this will cause the future utility to be lower. Thus, we seek to understand whether one of these two possibilities is correct, or whether there is a "sweet spot" where a certain number of historical thresholds is just enough for the mechanism to implicitly learn $\mathcal{T}_F$ but does not result in the privacy budget being spread too thin.

To empirically quantify the effect of both the threshold distribution concentration as well as historical workload size on `RAP`'s future utility, we evaluate `RAP` across a range of workload sizes using the three specified distributions over features in both the ADULT and LOANS datasets. To put `RAP`'s future utility into context, we also evaluate the future utility of the `All-0` baseline mechanism. Refer to Table 5 for a summary of this experiment.

Figure 12 shows the results of this experiment. As in our prior experiments, each point of the `RAP` line is taken to be where `RAP` achieves minimal *present error* across all combinations of T, K evaluated. The future error at this minimizing T, K pair is then evaluated and plotted, along with a corresponding 95% confidence interval to account for randomness both between independent repetitions and across sampling future thresholds from the threshold distribution. For real-world applications, this reflects what a practitioner using `RAP` would be able to do; i.e., choose

Primary Mechanism	RAP
Baseline Mechanism	All-0
Utility Measure	err_F
D	ADULT, LOANS
ϵ	0.1
δ	$1/\lvert D\rvert^2$
$\lvert W_H\rvert$	$1, 4, 16, 64, 256, 1024$
n'	1000
T	$1, 4, 16, 64$
K	$4, 16, 64, 256$
r	1
k	3
$\mathcal{T}_H, \mathcal{T}_F$	Uniform, Zipf, Geometric
γ	$0, 0.05, 0.1, 0.2, 0.5, 1$

Table 5: Experimental reference table for evaluating the future utility of RAP on r-of-k thresholds.

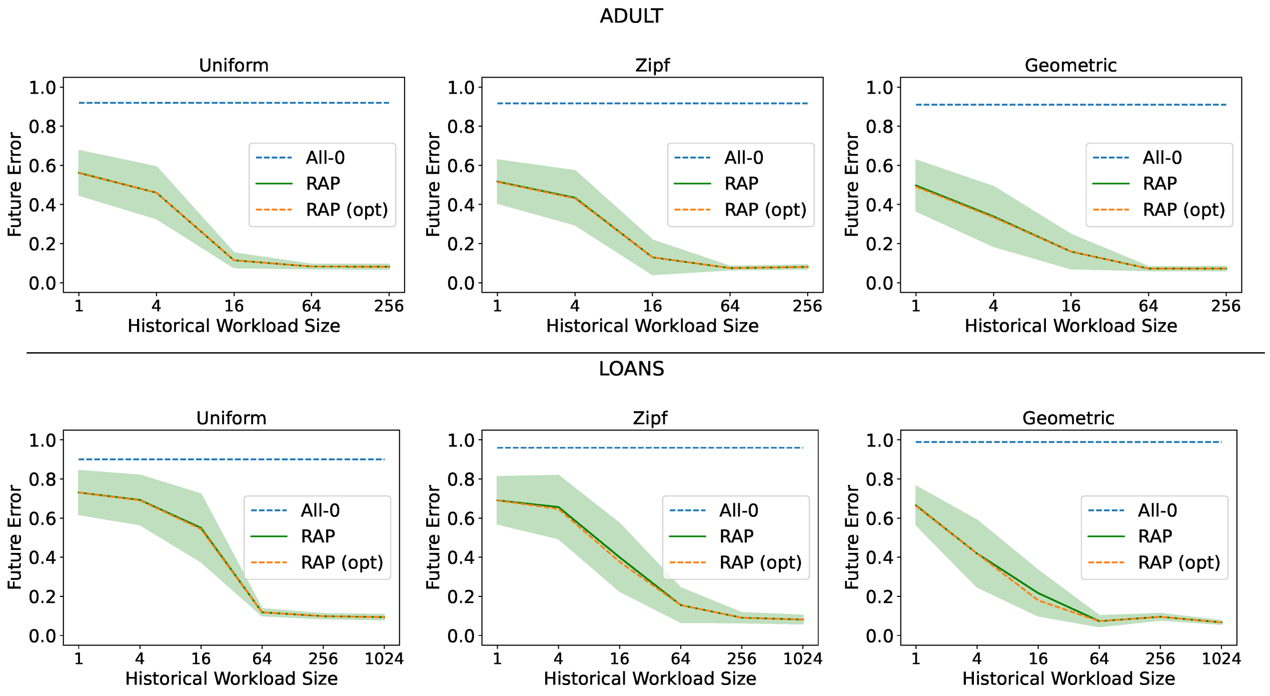

Figure 12: RAP's future error (and 95% confidence intervals) across all T, K values considered where RAP achieves minimal present error, plotted across a range of workload sizes and historical threshold distributions. "RAP (opt)" represents RAP's future error across all T, K values considered where RAP achieves minimal future error. Future error of All-0 included as a baseline.

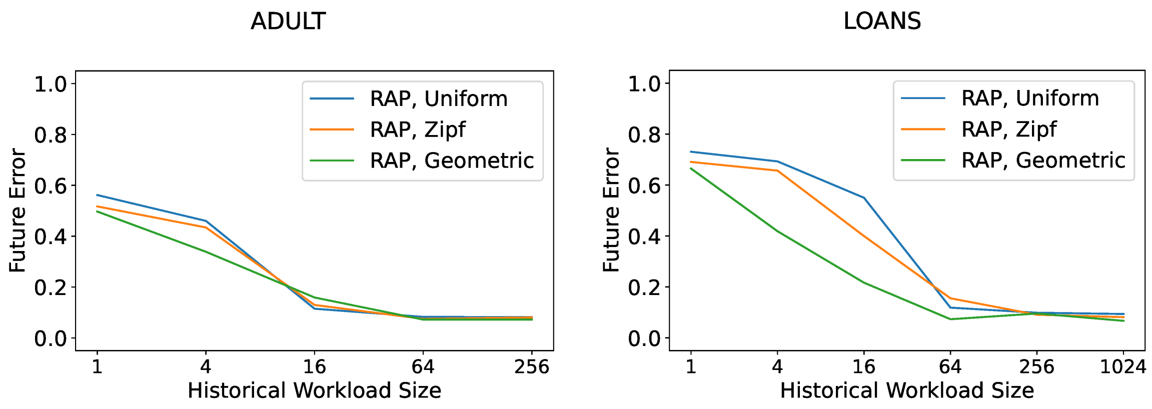

Figure 13: RAP's future utility on each threshold distribution across a range of workload sizes.

the best performing instance of RAP across T, K values on the present error metric (since they would not be able to evaluate future error), and then use that instance to answer future queries. Ideally though, the practitioner would have omnisciently been able to choose the best performing instance of RAP across T, K values on the future error metric directly, as this approach will never have larger future error than the former (feasible) approach. To understand whether there is a significant difference in the future error between these two scenarios, we additionally plot the latter as "RAP (opt)". For each distribution individually, we find the results are as expected. Namely, RAP's future error is always lower than the baseline mechanism All-0's future error, and RAP's future error decreases as the number of historical thresholds that it is given increases. Interestingly, we find no evidence that there is any point in which the number of historical thresholds given to RAP becomes "too large" and causes RAP's future error to begin increasing. Instead, we find that RAP benefits from being provided more historical thresholds when the historical workload size is small, and then eventually reaches a point of diminishing returns. Additionally, we find that the future error corresponding to the RAP instance that attains minimal present error across T, K values is nearly identical to the future error corresponding to the RAP instance that attains minimal future error across T, K values. This indicates that in practice, answering future queries using the RAP instance that achieved minimal present error across T, K values will likely also yield the minimal future error.

To better visualize the differences across distributions, RAP's future error lines are overlayed in Figure 13 for both the ADULT and LOANS datasets. From this, we see that the differences between RAP's future error across all three distributions are not as striking as one may expect, although for small historical workload sizes (less than 16 and 64 on the ADULT and LOANS datasets respectively) we find that the results roughly align with our intuition: the least concentrated (Uniform) distribution induces the highest future error, while the most concentrated (Geometric) distribution induces the lowest future error. These findings, taken together with those of Figure 12, yield a simple, useful insight into how to achieve low future error with RAP in practice. Specifically, if the size of the historical workload is small, a practitioner can simply augment it by adding uniformly randomly sampled thresholds from the space of all possible thresholds (regardless of what the underlying threshold distribution $\mathcal{T}_H$ is). In the worst case, RAP's future error will be essentially unaffected (if $|W_H|$ was already in the region where returns are diminishing); in the best case, RAP's future error will be reduced significantly.

5.3.2 Effect of "Overfitting" the Synthetic Dataset

When answering a prespecified set of queries using RAP, the goal in the relaxed projection step is to achieve as close to a global minima as possible. In fact, although such an achievement is unlikely in practice, Aydore et al.'s theoretical utility result relies on a global minima having been reached. However, when the goal is to learn a model that generalizes to unseen data, it is well known that optimizing the loss function to a global minima will lead to an extremely overfit model. In the exact partial knowledge setting where future utility is the metric of choice, we seek to determine whether a conceptually similar "overfitting" phenomena may be occurring when RAP uses the

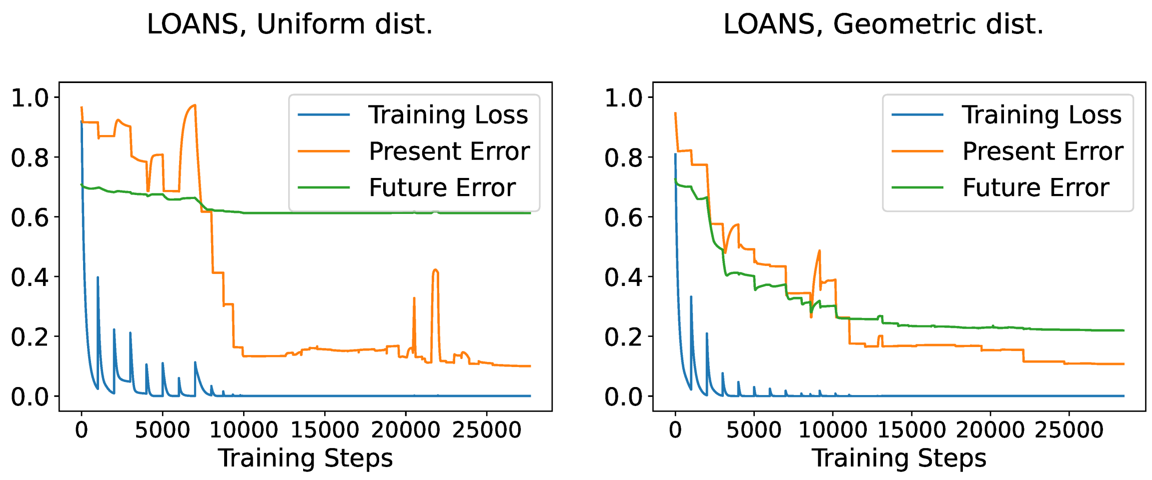

Figure 14: Training progress across iterations for RAP on Uniform vs. Geometric distributions over features in LOANS dataset, both with a small historical workload size of 4.

historical threshold workload to generalize to future thresholds.

Towards this, we recall our finding from Figure 12. Specifically, that RAP does not seem to noticeably overfit to the historical queries when selecting the adaptivity parameters T and K based on the instance of RAP that had minimal present error. However, this finding does not eliminate the possibility that RAP is overfitting to the historical queries during the synthetic dataset optimization procedure itself. For instance, in Figure 13 on the LOANS dataset at a historical workload size of 4, there is a significant difference between RAP's future errors on the Uniform vs. Geometric distributions. This could be explained either by RAP overfitting to the historical workload generated from the Uniform distribution (which is relatively less informative regarding which thresholds are likely to be sampled in the future), or it could simply indicate that the historical workload does not contain enough information about the relevant space of thresholds that RAP needs in order to generalize well. To analyze this possibility, we perform the same experiment as above while simultaneously evaluating RAP's future utility not just at the end of the optimization procedure, but after each iteration of the optimization procedure. Figure 14 displays the results, along with RAP's training loss and present error after each iteration of the optimization procedure. In classic ML, a canonical symptom of overfitting is observing a point in the training progress where the training error continues decreasing, but where the test error begins steadily increasing. In our setting, the analogue would be observing a point where the present error continues decreasing, but where the future error begins increasing. However, we do not observe such behavior in either graph, as the future error steadily decreases throughout the entire training procedure. The primary difference between the two graphs is that RAP's decrease in future error under the Uniform distribution is much smaller than under the Geometric distribution. This simply indicates that, as expected, RAP is able to take advantage of the significantly more informative (with respect to the relevant portions of the space future thresholds will be drawn from) historical workload from the Geometric distribution. Viewed differently, in the case of the Uniform distribution, RAP did not "overfit" to the historical workload — rather, the historical workload simply did not provide enough information to RAP about the relevant remainder of the query space.

The take-away from these findings is that while RAP would have benefited from having a larger historical threshold workload, it would not have benefited from introducing analogues to other classic overfitting remedies. For example, a practitioner may be tempted to reserve a held-out set of thresholds from the historical workload with the intention of using them between iterations as a proxy to estimate future utility, stopping the training early when the error on the held-out set begins increasing. Not only do these findings indicate that such a strategy would not be beneficial, but combined with the findings from the previous experiment, we conclude that such a strategy would result in relatively *greater* future error due to the reduced historical workload that RAP is given.

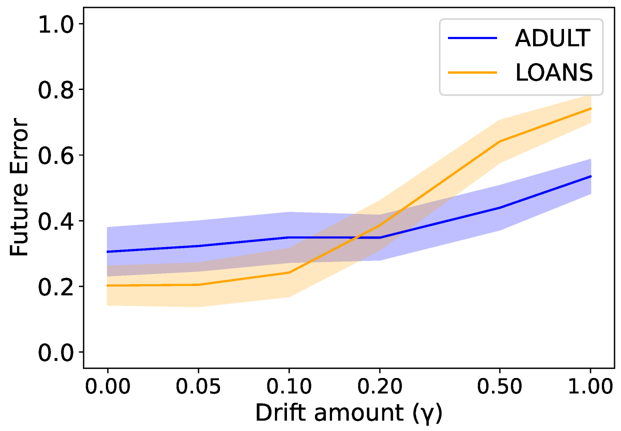

Figure 15: Future error of RAP across a range of range of distributional drift amounts on the ADULT and LOANS datasets, given small historical workload sizes of 4 and 16 respectively.

5.3.3 Effect of Threshold Distribution Drift

In the drifting partial knowledge setting, as the future features distribution $\mathcal{F}_F$ drifts further from the historical features distribution $\mathcal{F}_H$, it is clear that RAP's future utility should decrease. However, it is unclear how *sensitive* RAP's future utility is to such drift. Thus, we seek to quantify the extent to which RAP can tolerate distributional drift while maintaining high future utility.

To achieve this, we evaluate RAP's future utility in the following experiment. We first define the historical features distribution $\mathcal{F}_H$ using the aforementioned highly concentrated Geometric distribution over features in both the ADULT and LOANS datasets. We then measure RAP's future error across a range of drift amounts. Because RAP achieved low future error in the exact partial knowledge setting on all distributions when the workload size was large enough, we anticipate that distributional drift will similarly not have a significant impact when the historical workload size is large. Thus, in Figure 15, we evaluate the impact of distributional drift specifically with small historical workload sizes of 4 and 16 on the ADULT and LOANS datasets respectively.

The results of this experiment reveal that on both datasets, RAP is fairly impervious to distributional drift. RAP's future error only begins to exhibit a significant increase at approximately $\gamma = 0.4$ on the ADULT dataset and $\gamma = 0.1$ on the LOANS dataset. Comparing with Figure 11, these points both correspond to an expected total variation distance between the historical and future features distributions of approximately 0.5 on their respective datasets. Thus, we are able to conclude that even if the future features distribution drifts from the historical features distribution by a moderate amount, RAP can still be expected to maintain high utility.

6 Additional Related Works

In this section, in addition to the prior works on large-scale query answering previously discussed (Section 1.1), we discuss other important works related to differentially private query answering. We begin by discussing some works (concurrent with and subsequent to our work) related to answering large sets of prespecified queries. For the mechanisms defined in these works, a prime direction for future research would be to evaluate them analogously to our evaluation of RAP in this work. For instance, evaluating their scalability to larger query spaces as well as their generalizability for answering queries posed in the future, perhaps in a manner similar to Tao et al. [TMH$^+$21]). We then briefly discuss some lines of research related to the general problem and settings explored in this work.

Answering Many Queries

One closely related work to the goals of this work is that of Liu et al. [LVW21], which studies the problem of constructing an algorithmic framework for privately answering a prespecified set of statistical queries — our first setting of interest. Concretely, the framework they construct unifies several DP mechanisms which specifically answer queries by building a synthetic dataset through iterative, adaptive updates. These mechanisms include the previous practical state-of-the-art mechanisms [GAH$^+$14, VTB$^+$20], as well as a modified variant of a *preliminary* version of the RAP mechanism (where a softmax transformation [Bri90] is applied to each row of the synthetic dataset D after each iteration of RAP's optimization procedure). Liu et al. then leverage their framework to design two new mechanisms for answering prespecified sets of queries, and empirically show that both achieve high utility. However, in their empirical evaluations, Liu et al. find that the modified RAP mechanism's utility is on par with the utility of their two newly proposed mechanisms, and that RAP is computationally cheaper to execute. Thus, we do not additionally evaluate their two new mechanisms in this work. Moreover, Aydore et al. have subsequently updated the RAP mechanism to incorporate a similar modification (applying the Sparsemax transformation [MA16], and optionally finishing with randomized rounding) and showed that it further improves utility — in turn, further justifying our focus on the RAP mechanism.

Along similar lines, another closely related work is the recently introduced *Adaptive and Iterative Mechanism* (AIM) by McKenna et al. [MMSM22]. AIM is a mechanism for DP synthetic data generation to specifically answer workloads of marginal queries. The high-level idea of their approach is similar to that of RAP and Liu et al.'s work [LVW21], adaptively selecting marginals to use to optimize the synthetic dataset. However, their work takes this a step further by designing a method to more intelligently perform the selection. Moreover, they develop new techniques to quantify the uncertainty to answers derived from the generated synthetic data. Empirically evaluating AIM, they show that it generally outperforms prior state-of-the-art mechanisms, including RAP. However, their evaluation setting was somewhat different; specifically, they reduced the domain size of the datasets by discretizing numerical features into 32 equal-width bins. This makes the optimization problem significantly easier for all mechanisms they evaluate, which is highly useful when running a wide range of experiments across many random trials. However, it leaves AIM's utility unclear when the data is unbinned and sparse (e.g., for a numerical attribute with 100 possible values). Moreover, since the source code of AIM's implementation was never released, we consider a ground-up reimplementation of AIM amenable to large-scale evaluations on large and sparse data spaces to be out of the scope of this work. Performing such evaluations, especially in connection to the computational resources required by each method (AIM, RAP, and others), is a prime direction for future work.

Another closely related work is the concurrent theoretical work of Nikolov [Nik22], which proposes and analyzes a new mechanism for answering sets of prespecified queries with differential privacy. Their new mechanism is based on randomly projecting the queries to a lower dimensional space, answering the projected queries with a simple DP additive-noise mechanism, then lifting the answers back into their original dimension. The primary focus and contribution of their work is the thorough mathematical analysis of the mechanism's utility, showing that it achieves optimal worst case sample complexity under an average error metric. Such results are less directly relevant to our work though, as our focus is on different error metrics for fixed real-world datasets (rather than in the worst case across all possible datasets). However, conceptually, Nikolov's newly proposed mechanism could be used to tackle the same problem as our work. Practically though, the runtime of Nikolov's mechanism (although polynomial) would prevent it from being used to answer the large number of queries that we answer with RAP in this work. An intriguing direction for future work would be adapting Nikolov's new mechanism for practical query answering, and determining ways to scale it up to accurately answer queries on a truly large scale.

A final line of closely related work is the subsequent work of Vietri et al. [VAA$^+$22]. The focus of their work is explicitly on enhancing the RAP mechanism, creating a new mechanism they call RAP++. Their goal is orthogonal to the goal of this work, in that they seek to extend the original RAP mechanism so that it is able to support numerical features natively. Prior to their work, RAP required one-hot discretization of any numerical features in the dataset. For features with wide numerical ranges, one-hot discretization greatly increases the dimensionality of RAP's optimization problem, which in turn increases the computational burden and simultaneously decreases the mechanism's overall utility. In RAP++, Vietri et al. incorporate tempered sigmoid annealing and random linear projection queries into RAP in order to handle a mixture of categorical and numerical features without any discretization. They perform several empirical evaluations on RAP++, finding that it achieves state-of-the-art utility and runtime. Despite their goal being orthogonal to the goal of this work, our findings from this work could be used to further improve the RAP++ mechanism and its evaluation.

Related Lines of Research

One related (but disjoint) line of research is on the *public/private* model of differential privacy, where some data must be protected with differential privacy while the remaining "public" data requires no privacy protections [BNS13, JE13, HCB16, PAE+17, BCM+20, ABM19, LVS+21, TBM21]. These works have shown that mechanisms can be designed which make use of a small amount of public data in order to significantly boost utility. Our work differs from this model in that it does not directly make use of any public data. In our newly defined partial knowledge setting, we instead assume that the entire set of user data D is private, but that there exist publicly known historically posed queries Q_H which are not privacy sensitive. Assuming that Q_H was generated from a random distribution $\mathcal{T}_H$, we seek to understand the extent to which the RAP mechanism is able to take advantage of Q_H using D in order to accurately answer future queries generated from a distribution $\mathcal{T}_F$ related to $\mathcal{T}_H$.

The final related line of work is on reconstruction attacks, which studies how accurately sets of queries can be answered before private information in the dataset can be recovered. The high level results of this research can be summarized through the Fundamental Law of Information Recovery [DR+14]: "overly accurate answers to too many questions will destroy privacy in a spectacular way." Initial work on reconstruction attacks [DN03] inspired the conception of DP, and subsequent works have improved the computational efficiency of attacks, improved the theoretical analyses of attacks, or crafted highly effective attacks to specific cases [DMT07, DY08, MN12, DSSU17, GAM19]. Although somewhat related, the focus of this line of work significantly differs from the focus of our work. In research on reconstruction attacks, the basic goal is to find worst-case sets of queries (or the minimal sizes thereof) such that it is impossible to answer them all accurately while simultaneously maintaining privacy. In this work, our focus is not on generic worst-case queries, but rather on efficiently and accurately answering practical sets of prespecified or randomly sampled queries with privacy. Thus, the works on reconstruction attacks are not directly relevant to our problem in either of the two settings we consider.

7 Conclusions

In this work, we address the high-level research question: *to what extent are differentially private mechanisms able to answer a large number of statistical queries efficiently and with low error?* We analyze this problem in two settings, the classic prespecified queries setting, and a new setting that we introduced where only partial knowledge of the queries is available to the DP mechanism in advance. In both settings, our contributions are grounded in the state-of-the-art DP mechanism for answering large numbers of queries, the RAP mechanism. In the prespecified queries setting, we perform a focused but thorough reproducibility study on Aydore et al.'s original evaluation of RAP in order to clarify its value and strengthen its adoptability for practical uses. We also expand the class of queries that RAP is capable of evaluating, thus extending RAP's applicability in practice. Aside from the prespecified queries setting, we concretely specify a new partial knowledge setting where a mechanism is provided with a set of historically posed queries which are similar to queries that will be posed in the future. In this setting, we define a machine learning inspired utility measure to quantify a mechanism's ability to answer such future queries. Then, utilizing this utility measure, we evaluate RAP's suitability for generating synthetic datasets to answer queries posed in the future, finding that it is both efficient and effective. Our findings in this chapter further the state of the art in differentially private large-scale query answering, and additionally open new directions for future work on other problems in differential privacy within our newly defined partial knowledge setting.

Acknowledgements

This work was funded in part by a Privacy Enhancing Technologies Award[‖] from Facebook and NSF awards #1916153, #1956435, and #1943584.

[‖]https://research.facebook.com/research-award-recipients/year/2021/?s=korolova

References

[ABK$^+$21] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. Differentially private query release through adaptive projection. In *International Conference on Machine Learning*, pages 457–467. PMLR, 2021.

[ABM19] Noga Alon, Raef Bassily, and Shay Moran. Limits of private learning with access to public data. *Advances in neural information processing systems*, 2019.

[Abo18] John M Abowd. Staring-down the database reconstruction theorem. In *Joint Statistical Meetings, Vancouver, BC*, page 234, 2018.

[ACG$^+$16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[BCD$^+$07] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282, 2007.

[BCM$^+$20] Raef Bassily, Albert Cheu, Shay Moran, Aleksandar Nikolov, Jonathan Ullman, and Steven Wu. Private query release assisted by public data. In *International Conference on Machine Learning*, pages 695–703. PMLR, 2020.

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138, 2005.

[BFH$^+$18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[BLR08] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 609–618, 2008.

[BM06] Peter L Bartlett and Shahar Mendelson. Empirical minimization. *Probability theory and related fields*, 135(3):311–334, 2006.

[BNS13] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 363–378. Springer, 2013.

[Bri90] John S Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in neural information processing systems*, pages 211–217, 1990.

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[Bur16] UC Bureau. American community survey (acs). *The United States Census Bureau nd https://www. census. gov/programs-surveys/acs (accessed May 5, 2021)*, 2016.

[CCK$^+$16] Yiling Chen, Stephen Chong, Ian A Kash, Tal Moran, and Salil Vadhan. Truthful mechanisms for agents that value privacy. *ACM Transactions on Economics and Computation (TEAC)*, 4(3):1–30, 2016.

[CKKL12] Mahdi Cheraghchi, Adam Klivans, Pravesh Kothari, and Homin K Lee. Submodular functions are noise stable. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1586–1592. SIAM, 2012.

[CKS18] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146, 2018.

[CR21] Mark Cesar and Ryan Rogers. Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics. In *Algorithmic Learning Theory*, pages 421–457. PMLR, 2021.

[CRB22] Miranda Christ, Sarah Radway, and Steven M Bellovin. Differential privacy and swapping: Examining de-identification's impact on minority representation and privacy preservation in the us census. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1564–1564. IEEE Computer Society, 2022.

[CTUW14] Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 387–402, 2014.

[Dai22] Donna Daily. Disclosure avoidance protections for the american community survey, Dec 2022.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[DMT07] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 85–94, 2007.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[DNR$^+$09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[DR$^+$14] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[DR19] David Durfee and Ryan M Rogers. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.

[DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Symposium on Foundations of Computer Science (FOCS)*, pages 51–60. IEEE, 2010.

[DSSU17] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4:61–84, 2017.

[DY08] Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *Annual International Cryptology Conference*, pages 469–480. Springer, 2008.

[Fra10] Andrew Frank. UCI machine learning repository. *http://archive.ics.uci.edu/ml*, 2010.

[GAH$^+$14] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, 2014.

[GAM19] Simson Garfinkel, John M Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62(3):46–53, 2019.

[GH06] Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

[GHRU13] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. *SIAM Journal on Computing*, 42(4):1494–1520, 2013.

[GRU12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Theory of Cryptography Conference*, pages 339–356. Springer, 2012.

[HCB16] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *International Conference on Machine Learning*, pages 555–563. PMLR, 2016.

[HLM12] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012.

[Hol79] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[HR10] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 61–70. IEEE, 2010.

[HRS12] Moritz Hardt, Guy N Rothblum, and Rocco A Servedio. Private data release via learning thresholds. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 168–187. SIAM, 2012.

[HTFF09] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[HW04] Kohei Hatano and Osamu Watanabe. Learning r-of-k functions by boosting. In *International Conference on Algorithmic Learning Theory*, pages 114–126. Springer, 2004.

[JE13] Zhanglong Ji and Charles Elkan. Differential privacy based on importance weighting. *Machine learning*, 93(1):163–183, 2013.

[KLPV87] Michael Kearns, Ming Li, Leonard Pitt, and Leslie Valiant. On the learnability of boolean formulae. In *Proceedings of the nineteenth annual ACM symposium on theory of computing*, pages 285–295, 1987.

[Lit88] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[LMH$^+$15] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[LVS$^+$21] Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Steven Wu. Leveraging public data for practical private query release. In *International Conference on Machine Learning*, pages 6968–6977. PMLR, 2021.

[LVW21] Terrance Liu, Giuseppe Vietri, and Steven Z Wu. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems*, 34, 2021.

[MA16] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR, 2016.

[MC89] Gary L Miller and Bradley W Carroll. Modeling vertebrate dispersal distances: alternatives to the geometric distribution. *Ecology*, 70(4):977–986, 1989.

[MMHM18] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment*, 11(10), 2018.

[MMSM22] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *Proc. VLDB Endow.*, 15(11):2599–2612, sep 2022.

[MN12] Shanmugavelayutham Muthukrishnan and Aleksandar Nikolov. Optimal private halfspace counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1285–1292, 2012.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[MSM19] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444. PMLR, 2019.

[NBRS22] Michelle Nixon, Andres Barrientos, Jerome Reiter, and Aleksandra Slavkovic. A latent class modeling approach for differentially private synthetic data for contingency tables. *Journal of Privacy and Confidentiality*, 12(1), 2022.

[Nik22] Aleksandar Nikolov. Private query release via the johnson-lindenstrauss transform. *arXiv preprint arXiv:2208.07410*, 2022.

[NTS14] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[NTZ13] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the 45th annual ACM symposium on Theory of computing*, pages 351–360, 2013.

[OVL18] Kensuke Okada, Joachim Vandekerckhove, and Michael D Lee. Modeling when people quit: Bayesian censored geometric models with hierarchical and latent-mixture extensions. *Behavior research methods*, 50(1):406–415, 2018.

[PAE$^+$17] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Pap19] Nicolas Papernot. Machine learning at scale with differential privacy in {TensorFlow}. In *2019 {USENIX} Conference on Privacy Engineering Practice and Respect ({PEPR} 19)*, 2019.

[Rod21] Rolando A. Rodríguez. Disclosure avoidance and the american community survey. 2021 ACS Data Users Conference, 2021.

[RR10] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 765–774, 2010.

[SS18] Joshua Snoke and Aleksandra Slavković. pmse mechanism: differentially private synthetic data with maximal distributional similarity. In *International Conference on Privacy in Statistical Databases*. Springer, 2018.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[SVK21] Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34, 2021.

[TBM21] Yuchao Tao, Johes Bater, and Ashwin Machanavajjhala. Prior-aware distribution estimation for differential privacy. *arXiv preprint arXiv:2106.05131*, 2021.

[TMH$^+$21] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021.

[TUV12] Justin Thaler, Jonathan Ullman, and Salil Vadhan. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming*, pages 810–821. Springer, 2012.

[Ull13] Jonathan Robert Ullman. *Privacy and the Complexity of Simple Queries*. PhD thesis, 2013.

[Ull16] Jonathan Ullman. Answering nˆ2+o(1) counting queries with differential privacy is hard. *SIAM Journal on Computing*, 45(2):473–496, 2016.

[UV11] Jonathan Ullman and Salil Vadhan. Pcps and the hardness of generating private synthetic data. In *Theory of Cryptography Conference*, pages 400–416. Springer, 2011.

[VAA$^+$22] Giuseppe Vietri, Cedric Archambeau, Sergul Aydore, William Brown, Michael Kearns, Aaron Roth, Ankit Siva, Shuai Tang, and Zhiwei Steven Wu. Private synthetic data for multitask learning and marginal queries. *arXiv preprint arXiv:2209.07400*, 2022.

[Vap98] Vladimir Vapnik. The support vector method of function estimation. In *Nonlinear modeling*, pages 55–85. Springer, 1998.

[Vap99] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[VTB$^+$20] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Steven Wu. New oracle-efficient algorithms for private synthetic data release. In *International Conference on Machine Learning*, pages 9765–9774. PMLR, 2020.

[YCLZ04] Jeffrey Xu Yu, Zhihong Chong, Hongjun Lu, and Aoying Zhou. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In *VLDB*, volume 4, pages 204–215, 2004.

[YSS$^+$21] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

[ZBH$^+$21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[ZDCW12] Jianping Zeng, Jiangjiao Duan, Wenjun Cao, and Chengrong Wu. Topics modeling based on selective zipf distribution. *Expert Systems with Applications*, 39(7):6541–6546, 2012.

A Appendix

Deferred Regression Analysis Details

In this portion, we present the details of the setup and results for the regression analysis on the utility impact of filtering "large" marginals out of `RAP`'s evaluation.

Present Error vs. Workload Size

For this regression analysis on each dataset, we define the following regression variables:

- x_1, x_2: dummy variable encodings for the three levels of ϵ evaluated. I.e.,
  - $x_1 = x_2 = 0$ represents $\epsilon = 0.01$.
  - $x_1 = 1, x_2 = 0$ represents $\epsilon = 0.1$.
  - $x_1 = 0, x_2 = 1$ represents $\epsilon = 1$.
- x_3: logarithm of the workload size.
- x_4: indicator variable representing whether thresholding was applied. I.e., $x_4 = 0$ if thresholding was not applied, $x_4 = 1$ if it was.
- ζ: stochasticity in the process (e.g., from randomness in the RAP mechanism due to privacy, from randomness in the marginal selection process across independent trials, etc.).

With these variables defined, we state the full regression model with interactions as

$$\text{err}_P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + (\beta_3 + \beta_4 x_1 + \beta_5 x_2)x_3 + (\beta_6 + \beta_7 x_1 + \beta_8 x_2 + (\beta_9 + \beta_{10} x_1 + \beta_{11} x_2)x_3)x_4 + \zeta,$$

and the restricted regression model as

$$\text{err}_P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + (\beta_3 + \beta_4 x_1 + \beta_5 x_2)x_3 + \zeta.$$

We then fit both the full and restricted regression models to the results of the RAP evaluations for the ADULT and LOANS datasets (separately). Regression results for the full models (ADULT on left and LOANS on right) are stated below.

Dep. Variable:	present_err	**R-squared:**	0.963
Model:	OLS	**Adj. R-squared:**	0.959
Method:	Least Squares	**F-statistic:**	266.6
Covariance Type:	nonrobust	**Prob (F-statistic):**	7.40e-76
No. Observations:	126	**Log-Likelihood:**	295.45
Df Residuals:	114	**AIC:**	-566.9
Df Model:	11	**BIC:**	-532.9

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
β_0	0.0320	0.009	3.415	0.001	0.013	0.051
β_1	-0.0066	0.013	-0.495	0.621	-0.033	0.020
β_2	-0.0248	0.013	-1.869	0.064	-0.051	0.001
β_3	0.0650	0.003	24.536	0.000	0.060	0.070
β_4	-0.0528	0.004	-14.075	0.000	-0.060	-0.045
β_5	-0.0600	0.004	-16.015	0.000	-0.067	-0.053
β_6	0.0280	0.013	2.120	0.036	0.002	0.054
β_7	-0.0309	0.019	-1.649	0.102	-0.068	0.006
β_8	-0.0277	0.019	-1.482	0.141	-0.065	0.009
β_9	-0.0036	0.004	-0.952	0.343	-0.011	0.004
β_{10}	0.0052	0.005	0.988	0.325	-0.005	0.016
β_{11}	0.0040	0.005	0.762	0.448	-0.006	0.014

Omnibus:	24.270	**Durbin-Watson:**	1.693
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	114.122
Skew:	0.434	**Prob(JB):**	1.65e-25
Kurtosis:	7.581	**Cond. No.**	64.4

Dep. Variable:	present_err	**R-squared:**	0.942
Model:	OLS	**Adj. R-squared:**	0.937
Method:	Least Squares	**F-statistic:**	193.4
Covariance Type:	nonrobust	**Prob (F-statistic):**	1.05e-75
No. Observations:	144	**Log-Likelihood:**	228.17
Df Residuals:	132	**AIC:**	-432.3
Df Model:	11	**BIC:**	-396.7

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
β_0	0.0372	0.019	1.982	0.050	7.32e-05	0.074
β_1	-0.0113	0.027	-0.425	0.671	-0.064	0.041
β_2	-0.0282	0.027	-1.062	0.290	-0.081	0.024
β_3	0.0966	0.004	21.626	0.000	0.088	0.105
β_4	-0.0767	0.006	-12.134	0.000	-0.089	-0.064
β_5	-0.0882	0.006	-13.953	0.000	-0.101	-0.076
β_6	0.0215	0.027	0.812	0.418	-0.031	0.074
β_7	-0.0273	0.038	-0.729	0.467	-0.102	0.047
β_8	-0.0275	0.038	-0.733	0.465	-0.102	0.047
β_9	-0.0039	0.006	-0.619	0.537	-0.016	0.009
β_{10}	0.0039	0.009	0.437	0.663	-0.014	0.022
β_{11}	0.0051	0.009	0.574	0.567	-0.013	0.023

Omnibus:	29.738	**Durbin-Watson:**	2.677
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	208.504
Skew:	0.355	**Prob(JB):**	5.29e-46
Kurtosis:	8.852	**Cond. No.**	75.6

Present Error vs. Number of Queries

For this regression analysis on each dataset, we define the same variables as in before, with the only change being that x_3 now represents the logarithm of the total number of consistent queries that RAP evaluates (rather than the size of the workload that RAP evaluates). With these variables, we define the same full and restricted regression models as before, and we fit both to the results of the RAP evaluations. Regression results for the full models (ADULT on left and LOANS on right) are stated below.

Dep. Variable:	present_err	**R-squared:**	0.889
Model:	OLS	**Adj. R-squared:**	0.879
Method:	Least Squares	**F-statistic:**	83.19
Covariance Type:	nonrobust	**Prob (F-statistic):**	3.83e-49
No. Observations:	126	**Log-Likelihood:**	227.07
Df Residuals:	114	**AIC:**	-430.1
Df Model:	11	**BIC:**	-396.1

	coef	**std err**	**t**	**P$>|t|$**	**[0.025**	**0.975]**
β_0	-0.3210	0.043	-7.438	0.000	-0.406	-0.235
β_1	0.2882	0.061	4.722	0.000	0.167	0.409
β_2	0.3014	0.061	4.939	0.000	0.181	0.422
β_3	0.0472	0.004	12.856	0.000	0.040	0.054
β_4	-0.0390	0.005	-7.516	0.000	-0.049	-0.029
β_5	-0.0436	0.005	-8.398	0.000	-0.054	-0.033
β_6	0.1198	0.057	2.110	0.037	0.007	0.232
β_7	-0.1237	0.080	-1.540	0.126	-0.283	0.035
β_8	-0.1189	0.080	-1.480	0.142	-0.278	0.040
β_9	-0.0127	0.005	-2.742	0.007	-0.022	-0.004
β_{10}	0.0123	0.007	1.886	0.062	-0.001	0.025
β_{11}	0.0124	0.007	1.894	0.061	-0.001	0.025

Omnibus:	53.796	**Durbin-Watson:**	1.512
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	189.737
Skew:	-1.528	**Prob(JB):**	6.30e-42
Kurtosis:	8.177	**Cond. No.**	572.

Dep. Variable:	present_err	**R-squared:**	0.887
Model:	OLS	**Adj. R-squared:**	0.877
Method:	Least Squares	**F-statistic:**	93.96
Covariance Type:	nonrobust	**Prob (F-statistic):**	7.68e-57
No. Observations:	144	**Log-Likelihood:**	180.50
Df Residuals:	132	**AIC:**	-337.0
Df Model:	11	**BIC:**	-301.4

	coef	**std err**	**t**	**P$>|t|$**	**[0.025**	**0.975]**
β_0	-0.6398	0.070	-9.171	0.000	-0.778	-0.502
β_1	0.5254	0.099	5.326	0.000	0.330	0.721
β_2	0.5873	0.099	5.952	0.000	0.392	0.782
β_3	0.0779	0.005	14.839	0.000	0.068	0.088
β_4	-0.0618	0.007	-8.321	0.000	-0.076	-0.047
β_5	-0.0709	0.007	-9.550	0.000	-0.086	-0.056
β_6	0.1453	0.096	1.509	0.134	-0.045	0.336
β_7	-0.1293	0.136	-0.949	0.344	-0.399	0.140
β_8	-0.1474	0.136	-1.082	0.281	-0.417	0.122
β_9	-0.0240	0.007	-3.647	0.000	-0.037	-0.011
β_{10}	0.0195	0.009	2.088	0.039	0.001	0.038
β_{11}	0.0227	0.009	2.431	0.016	0.004	0.041

Omnibus:	18.588	**Durbin-Watson:**	1.775
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	78.893
Skew:	-0.142	**Prob(JB):**	7.39e-18
Kurtosis:	6.615	**Cond. No.**	726.