Prometheus: Differential Privacy in the OMOP CDM

Tim Bergquist   Pascal Brandt

Department of Biomedical Informatics and Medical Education
University of Washington
Seattle, WA
{trberg, psbrandt}@uw.edu

ABSTRACT

In this work we investigate the application of differential privacy to electronic healthcare data stored in the OMOP Common Data Model schema. We investigate the feasibility of implementing differential privacy mechanism for counting queries, and discuss the limitations of the application of differential privacy in the context of healthcare datasets. We additionally consider the question of how to minimize the privacy budget ϵ while maintaining statistically meaningful analysis results.

Keywords

Differential privacy; healthcare data; OMOP CDM

1. INTRODUCTION

Statistical analysis of clinical data is fundamental for deriving novel medical insights; however, clinical data is highly sensitive and is protected by a variety of laws and institutional policies. While they protect patients from identification, these policies serve as major barriers to the widespread use and analysis of clinical data. Only a select few researchers have access to these valuable datasets, and even these individuals must go to great lengths to access these data by applying for permission from institutional review boards. While this can be frustrating, the risks involved for an individual if their data were to be obtained by a bad actor could lead to severe negative consequences for the database participant, such as increased insurance premiums or prejudice due to stigma associated with a medical condition. To make matters worse, even if seemingly anonymized data is released, these data could be combined with other available information about the individual in order to re-identify them.

In order to protect the privacy of the patients that comprise the database participants, it is necessary to apply privacy measures that are underpinned by rigorous mathematical theory. One such measure is *differential privacy*, which has the goal of allowing an analyst to perform statistically accurate analyses on a dataset, while simultaneously protecting the identities of the dataset participants [3]. The goal of applying these privacy measures is to lower the barrier to accessing these datasets, thereby enabling a larger community of researchers to conduct analyses with the hope of accelerating the rate of innovation in biomedical research.

CSE544 2018 Seattle, Washington USA

© 2018 ACM. ISBN 123-4567-24-567/08/06. . . $0.99

DOI: 10.475/123_4

2. PRIVACY

The term privacy has different meanings depending on who you ask and the given context. Colloquially, in the context of healthcare data, privacy is usually understood to mean that the individual has the agency to control who has the right to see their personally *identifiable* health information. A number of mechanisms have been tried to enforce this somewhat nebulous definition of privacy. The most well known of these methods is known as data anonymization, which simple attempts to remove any data elements that may be considered *identifiable*, such as names, telephone and social security numbers, addresses, etc. The problem with this technique is that it completely discounts the effect of using auxiliary information to triangulate the identity of an individual. The most famous example of this type of re-identification is the Massachusetts governor who was identified by combining voter records with insurance data [10]. Other examples include high profile companies such as AOL and Netflix releasing datasets that were subsequently shown to be re-identifiable [3].

In light of the glaring inadequacy of data anonymization and other methods to protect the privacy of database participants, we need a technique that can guarantee data privacy with some degree of mathematical rigor. We must therefore adopt a definition of privacy that can be expressed formally, and must take any current and future auxiliary information into account.

2.1 Differential Privacy

The term *differential privacy* was coined by Dwork in 2006 [2], but the ideas were first published in a paper by Dwork *et al* earlier in the same year [4]. One of the key insights of differential privacy, is that it attempts to ensure that anything that can be learned about an individual is independent of whether that individual is a participant in the database or not. This is formalized by considering the application of a privacy mechanism $\mathcal{K}$ to two datasets D and D' that differ in at most one row. In the healthcare context, this can be considered equivalent to removing (or adding) the data belonging to a single individual patient to (or from) the database.

2.1.1 Definition

A randomized function $\mathcal{K}$ (the privacy *mechanism*) is said to be ϵ-differentially private if it meets the following criterion.

$$Pr[\mathcal{K}(D) \in S] \leq exp(\epsilon) \times Pr[\mathcal{K}(D') \in S] \tag{1}$$

Intuitively, this means that the probability that any individual is a member of database D is less than or equal to their probability of being a member of database D' multiplied by a very small value (e^{ϵ}). This applies to any two databases D and D' that differ in at most one row. The function $\mathcal{K}$ therefore obscures the individual's membership in the database.

2.1.2 Sensitivity

```
SELECT
  COUNT(*)
FROM
  patient
WHERE
  patient.condition = "Diabetes"
```

Listing 1: Simple count of patients with diabetes.

One more definition is required before implementation is possible. If f is a database query (i.e. $f : D \to \mathbb{R}^d$, where d is the dimension of the returned result), then the sensitivity Δf is defined as:

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1 \tag{2}$$

The sensitivity quantifies how much the result of any given query can differ on any two databases D and D' that differ in at most one row. Consider as an example the query shown in listing 1, which returns just a single number. The sensitivity of this query is 1, since the result can differ by at most one (and in one position) on two databases that differ in at most one row. That is, if one patient is added or removed from the database, the result of the query in listing 1 can differ by at most one. Sensitivity is an important consideration when selecting the random function $\mathcal{K}$ in equation 1.

2.2 Privacy Budget

Every time a query result is returned, even if differential privacy is applied, some information about the underlying database is leaked. For this reason, the value of ϵ in equation 1 should actually be thought of as the total privacy budget. To indicate this, we use the notation ϵ_{total}. To illustrate how the privacy budget may be exploited, consider the database attack shown in figure 1. Each histogram shows the results of the query in listing 2 run 1000 times with different privacy budgets. The the goal of this attack is to uncover the exact number of females in the database. We see that as the privacy budget increases, the histogram becomes more concentrated around the exact value (64347), and when $\epsilon_{total} = 100$, there is an obvious peak at exactly this value.

The selection of this privacy budget is a social question, but values such as 0.01, 0.1, $ln2$ or $ln3$ have been suggested [3]. In this work we will investigate the effect that the value of ϵ_{total} has on the statistical significance of the results of various clinically relevant statistical analyses.

```
SELECT
  COUNT(*)
FROM
  patient
WHERE
  patient.gender = "Female"
```

Listing 2: Simple count of female patients.

2.3 Mechanisms

One of the challenges of implementing differential privacy is choosing the privacy mechanism, which is the random function $\mathcal{K}$ in equation 1. The simplest privacy mechanism that has been shown to be effective simply adds noise drawn from the Laplace distribution, symmetric about zero and scaled by $\Delta f/\epsilon$:

$$noise = Lap(\Delta f/\epsilon) \tag{3}$$

Adding an independently generated noise term to each row of the query result ensures ϵ-differential privacy. For example, in the case where the query result is just a single numeric value, we simply add $Lap(1/\epsilon)$ to this value.

An additional privacy mechanism, not explored in this work, is known as the exponential mechanism [9], which actually describes how to design algorithms that preserve privacy as well as other data properties such as covariances between attributes. The technique has been successfully applied to generate synthetic datasets that are statistically similar to the underlying data [5], but that is beyond the scope of our work.

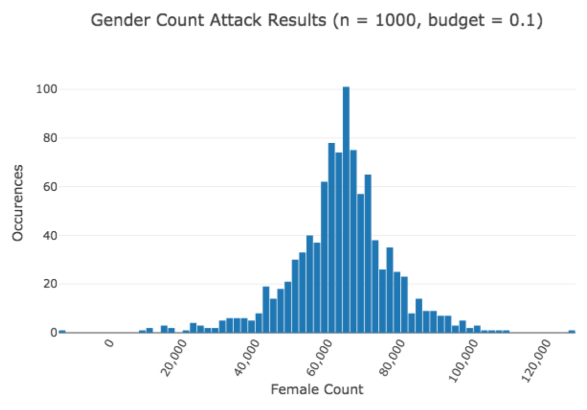
3. DATASET

The dataset used for the analyses presented in this work is the Centers for Medicare & Medicaid (CMS) Entrepreneurs' Synthetic Public Use File (DE-SynPUF) dataset[1]. CMS is part of the U.S. Department of Health and Human Services, and is responsible for the oversight of federal healthcare programs, including those involving health information technology. CMS is also responsible for administration of the Health Insurance Portability and Accountability Act (HIPAA), a significant portion of which deals with privacy of protected health information (PHI).

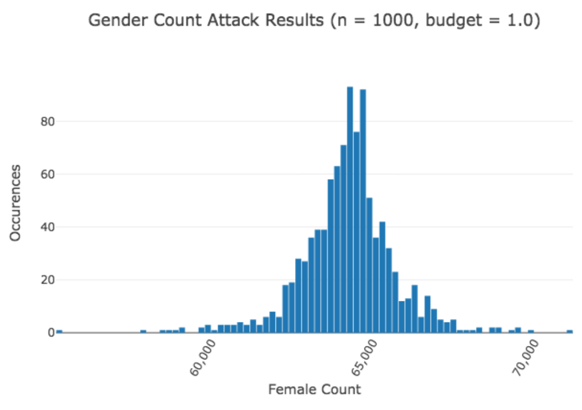
The goal of the SynPUF dataset is to provide "*a realistic set of claims data in the public domain while providing the very highest degree of protection to the Medicare beneficiaries' protected health information.*" The data was constructed by considering a 5% random sample from Medicare beneficiaries in 2008 and the claims from 2008 to 2010. While the claims data represent real Medicare claims, beneficiaries in the dataset are entirely synthetic, and identified only by an assigned synthetic identifier. The potential for empirical research is therefore limited; however, the data are adequate for the purposes of demonstrating the application of differential privacy.

For this project we consider only a subset of 5% of the DE-SynPUF dataset, which corresponds to about 166k patients,

[1]DE-SynPUF is freely available on the CMS website.



(a) $\epsilon_{total} = 0.1$



(b) $\epsilon_{total} = 1$

Figure 1: Gender attacks

3k locations, 28M conditions, and the record of roughly 27M procedures. Once the dataset and vocabularies are loaded, and indices are created, the resulting PostgreSQL data files are over 16GB in size.

3.1 Schema

The Observational Health Data Sciences and Informatics (OHDSI) consortium[2] is an interdisciplinary, multi-stakeholder collaboration of researchers in the biomedical domain [6]. One of the outputs of this collaboration is the so-called Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM), which is a collectively established standardized relational database schema used to store various types of electronic health data, including both clinical and health economics (insurance claims) data. The schema[3] contains 40 tables, and 69 indices are created in addition to the primary key indices to improve query execution time.

3.2 Translation

Since the CMS dataset is not released in the OMOP CDM format, it is necessary to do some data transformation and loading (ETL). Fortunately, since the dataset has broad appeal, a working group within the OHDSI consortium has published some Python scripts[4] to assist in this process. Using these scripts, as well as manually downloading and importing the required clinical vocabularies, we were able to construct our research database.

4. APPROACH

We began by loading the database as described above. We then implemented the Laplace noise mechanism applied to a query that returns only a single numeric value. We demonstrate that we can protect the privacy of individuals in a database from identification by using Laplacian noise. We then go on to evaluate the effects of the added noise to the integrity of the data by comparing the original dataset to the perturbed data.

[2]https://www.ohdsi.org/
[3]DDL, constraints, and indices available on GitHub here.
[4]https://github.com/OHDSI/ETL-CMS

The data loading steps, as well as all the Jupyter notebooks can be found in the GitLab repository[5].

5. EVALUATION

5.1 Demonstrating Simple Differential Privacy

5.1.1 *Dataset Statistics*

In this evaluation, we look at the statistics of diabetic patients in dataset. We define a diabetic patient as any patient who has had a least one clinical finding with the string “diabetes” in the condition description. Based on this definition, there are a total of 605 possible conditions that will define a patient as diabetic.

Using these definitions, we identified 81,051 patients who were diabetic and 17433 patients who were not diabetic.

5.1.2 *Simple Counting Attack*

We started with the simplest possible implementation. Say we want to find out something about patient whose id = 2 but we are only allowed to ask for counts. We can use the query in listing 3 to ask the database for counts of unique patient ids who are diabetic while also excluding patients with id = 2. The query returns a count of 81,050, one less than our diabetic population. We now know that patient id = 2 has diabetes. Information has leaked.

5.1.3 *Protecting the Individual*

By perturbing the return counts by a random value from our afore mentioned Laplace distribution, we can protect the identity of individuals within our database. As an example, take the query in listing 4. We can use so many columns in the demographics table, that the results begin returning single individuals like in table 1.

In order to ensure a privacy of epsilon 1, we use the Laplace noise function with epsilon = 1, and perturb the results by the random noise. In table 2, we see that enough noise has been added to the count numbers to protect individual patients.

[5]https://gitlab.cs.washington.edu/psbrandt/differential-privacy

```
SELECT
  COUNT( DISTINCT person_id )
FROM
  condition_occurrence
WHERE
  condition_concept_id IN(
    SELECT
      concept_id
    FROM
      concept
    WHERE
      (
        concept_name LIKE '%Diabetes%'
        OR concept_name LIKE '%diabetes%'
      )
      AND domain_id = 'Condition'
      AND concept_class_id = 'Clinical Finding'
  ) AND person_id != 2;
```

Listing 3: Example of deducing the clinical status of patients through query counts.

```
SELECT
  p.gender_concept_id,
  p.year_of_birth,
  p.month_of_birth,
  p.day_of_birth,
  COUNT(*)
FROM
  person p
GROUP BY
  p.year_of_birth,
  p.gender_concept_id,
  p.month_of_birth,
  p.day_of_birth,
  p.time_of_birth;
```

Listing 4: Example of deducing the clinical status of patients through query counts.

5.2 The Trade-off: Privacy vs Usability

When deciding how low epsilon should be when calculating the scale of the Laplacian noise, we need to take into account the effect the noise will have on the integrity of the data. Adding too much noise will make the data useless for meaningful statistical analysis, but adding too little will leave the people in the dataset vulnerable to identification.

5.2.1 *Evaluation Query*

In this evaluation, we are looking at the effects of perturbation on a distribution. The distribution we will be looking at is the histogram of ages of diabetic patients. We used the Query in listing 5 to collect the counts of the ages in the diabetics population. In figure 2, the age distribution is shown. This is the distribution that we will be perturbing and using as our data integrity benchmark.

5.2.2 *Usability Evaluation*

We evaluated the effect of a decreasing epsilon on the integrity of the data by comparing the distribution of the original, unperturbed data to that of the perturbed data at different values of epsilon. We measured distribution similarity using two different tests, a Pearson's correlation and Student's T-test.

```
SELECT
  ages.age,
  COUNT(*)
FROM (
  SELECT
    p.person_id,
    min((EXTRACT
    (YEAR from con.condition_start_date))
      - p.year_of_birth) as age
  FROM
    concept c,
    person p,
    condition_occurrence con
  WHERE
    con.person_id=p.person_id AND
    con.condition_concept_id=c.concept_id AND
    (c.concept_name LIKE '%Diabetes%' OR
     c.concept_name LIKE '%diabetes%') AND
     c.domain_id = 'Condition' AND
     c.concept_class_id = 'Clinical Finding'
  GROUP BY
    p.person_id) as ages
  GROUP BY ages.age;
```

Listing 5: Query for collecting the counts of the ages for all diabetic patients.

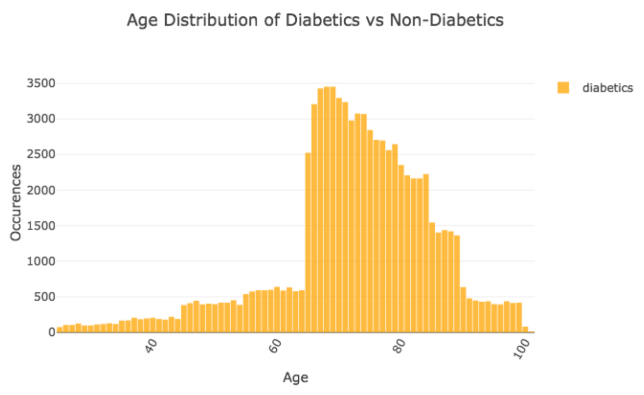


Figure 2: Age distribution of diabetics patients. This distribution will be used as our comparison benchmark

Starting with a value of epsilon 10, for each step of the comparison, we decreased the epsilon by a factor of 10 until we finally evaluated epsilon 0.001 in the case of the T-test and epsilon 0.0001 in the Pearson's correlation. For each epsilon, we ran 50 simulations, perturbing the data with Laplacian noise at a scale of 1/epsilon.

In figure 3, we see that as epsilon decreases, the p-value of the T-tests drops exponentially, crossing the significance threshold at epsilon=0.01. It is interesting to note that many publications recommend an epsilon of 0.01. However, according to this evaluation, you would run the risk of having significantly different data than your original dataset, opening the possibility of deriving false conclusion from a statistical analysis.

In figure 4, we see the same result; as epsilon decreases, the p-value of the Pearson's correlation drops exponentially, crossing the significance threshold at an epsilon value of 0.0001. Notice that the Pearson's correlation is far more

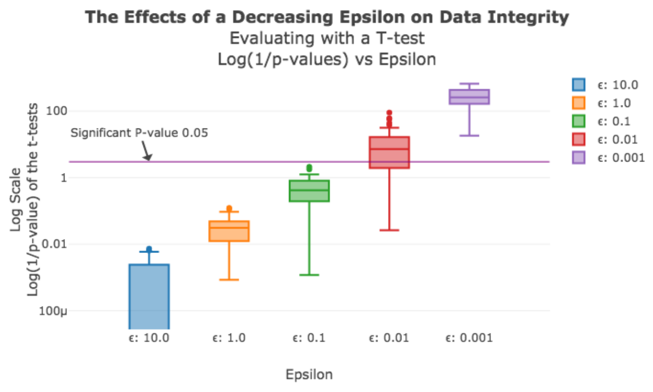


Figure 3: Log(1/p-values) vs Epsilon, where the p-value is derived from the T-test comparing the original dataset to the perturbed.

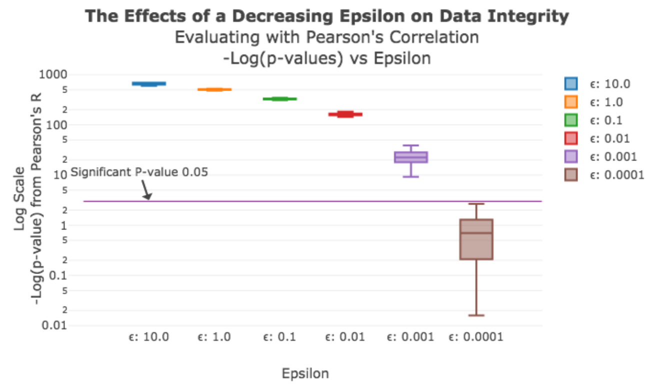


Figure 4: -Log(p-values) vs Epsilon, where the p-value is derived from the T-test comparing the original dataset to the perturbed.

forgiving that the Student's t-test. According to this evaluation, you could safely perturb your data with Laplacian noise generated using an epsilon of 0.01. You could follow the recommendations of the privacy community while maintaining confidence in your statistical findings.

The Pearson's correlation evaluation is probably more accurate in its assessment. The Student's t-test is more susceptible to changes in the distribution outliers. Since the simulations are going to be adding a lot of noise to the tail ends of the distributions at low levels of epsilon, the t-test will probably over estimate the effects of the data integrity.

Table 1: A selection of results from query 4 demonstrating a narrowing down of individuals by using a large number of variables.

gender_id	birth_year	birth_month	birth_day	count
8532	1974	4	1	2
8507	1979	6	1	1
8532	1982	7	1	1

5.3 Differentially Private Contingency Tables

We have shown that differential privacy is useful for maintaining the privacy of the individuals in our dataset while maintaining the usability of the data for analysis. Using these principles we generated five contingency tables combining different variables, asking medically relevant questions.

Table 2: A selection of results from query 4 that have been perturbed using random noise from a Laplace distribution.

gender_id	birth_year	birth_month	birth_day	count
8532	1974	4	1	5
8507	1979	6	1	2
8532	1982	7	1	2

```sql
SELECT EXTRACT(
  'year' FROM date_trunc('decade',
    make_date(p.year_of_birth,
      p.month_of_birth,
      p.day_of_birth))) AS
        birth_decade,
  CASE p.gender_concept_id
    WHEN 8532 THEN 'Female'
    ELSE 'Male' END AS gender,
  COUNT(*)
FROM
  condition_occurrence co,
  person p
WHERE
  co.person_id = p.person_id AND
  condition_concept_id = '4241530' -- HIV
GROUP BY
  date_trunc('decade', make_date(
    p.year_of_birth,
    p.month_of_birth,
    p.day_of_birth)),
  p.gender_concept_id
ORDER BY
  birth_decade,
  gender;
```

Listing 6: SQL query for finding all HIV patients by year and gender.

The GROUP BY queries we ran were:

1. Find all patients who have a Heart Failure condition and GROUP BY Gender and the decade in which they were born. Listing 10 is the query used.

2. Find all patients who have HIV and GROUP BY Gender and the decade in which they were born. Listing 6 is the query used.

3. Find all patients who have Diabetes and GROUP BY Gender and the decade in which they were born. Listing 7 is the query used.

4. Find all patients who have used the top eight most used drugs and GROUP BY Gender. Listing 8 is the query used.

5. Calculate the average length of hospital stay (days) for each patient and GROUP BY their average length of stay. Listing 9 is the query used.

5.3.1 *Contingency Tables*

For each of the queries, we used epsilon 0.1 to perturb the counts. We also compared each perturbed table to the original query results without perturbation analyzing the cosine

Table 3: A contingency table with differentially private counts for Heart Failure patients broken down by the decade of Birth and Gender

Gender Birth Decade	Female	Male
1900.0	2991	1156
1910.0	33325	14695
1920.0	99338	66047
1930.0	113757	93855
1940.0	59244	52999
1950.0	18198	18262
1960.0	11278	10328
1970.0	4556	4944
1980.0	1386	1129

Table 4: A contingency table with differentially private counts for HIV broken down by the decade of Birth and Gender

Gender Birth Decade	Female	Male
1900.0	31.0	NaN
1910.0	152.0	68.0
1920.0	468.0	263.0
1930.0	602.0	468.0
1940.0	374.0	260.0
1950.0	112.0	160.0
1960.0	156.0	95.0
1970.0	34.0	61.0
1980.0	17.0	19.0

similarity of the original matrix to the unperturbed matrix. Table 3 is the perturbed results from the listing 10 query for heart failure patients. The cosine similarity between the perturbed table and the original table is 0.99999997242. Table 4 is the perturbed results from the listing 6 query for HIV failure patients. The cosine similarity between the perturbed table and the original table is 0.999832337593. Table 5 is the perturbed results from the listing 7 query for Diabetic patients. The cosine similarity between the perturbed table and the original table is 0.999999998181. Table 6 is the perturbed results from the listing query 8 for drug use by gender. The cosine similarity between the perturbed table and the original table is 0.999999814847. Table 7 is the perturbed results from the listing query 9 for average length of stay per patient. The cosine similarity between the perturbed table and the original table is 1.0. Cosine similarity results are summarized in Table 8.

Table 5: A contingency table with differentially private counts for Diabetes patients broken down by the decade of Birth and Gender

Gender Birth Decade	Female	Male
1900.0	7469	2322
1910.0	83017	32368
1920.0	259309	163332
1930.0	336061	260821
1940.0	181434	154883
1950.0	50172	48176
1960.0	29634	29408
1970.0	14726	13798
1980.0	4285	3990

```sql
SELECT EXTRACT('year' FROM date_trunc('decade',
  make_date(p.year_of_birth,
    p.month_of_birth,
    p.day_of_birth))) AS birth_decade,
  CASE p.gender_concept_id WHEN 8532 THEN 'Female'
   ELSE 'Male' END AS gender,
  COUNT(*)
FROM
  person p,
  condition_occurrence con_oc,
  concept con
WHERE
  p.person_id = con_oc.person_id AND
  con_oc.condition_concept_id = con.concept_id AND
  con.concept_class_id = 'Clinical Finding' AND
  con_oc.condition_concept_id = con.concept_id AND
  ( con.concept_name LIKE '%Diabetes%' OR
    con.concept_name LIKE '%diabetes%' ) AND
  con.domain_id = 'Condition'
GROUP BY
  date_trunc('decade', make_date(p.year_of_birth,
    p.month_of_birth,
    p.day_of_birth)),
  p.gender_concept_id
ORDER BY
  birth_decade,
  gender
```

Listing 7: SQL query for diabetes by birth decade.

Table 6: A contingency table with differentially private counts for the number of patients who take the top 8 most used drugs broken down by the gender of the patients

Gender Drug Name	Female	Male
Epoetin Alfa	83343	55904
Gemfibrozil 600 MG	22434	14955
Influenza virus vaccine	43863	33562
Lovastatin 20 MG	19404	12682
Omeprazole 20 MG	22328	13834
Oxygen 99 % Gas	37517	23965
Simvastatin 40 MG	17658	11723
paricalcitol	45418	30434

6. RELATED WORK

The first notable implementation of differential privacy is the *Privacy Integrated Queries* (PINQ) framework [8]. This framework, developed by one of the authors of the original differential privacy paper, exposes an API that can be used for differential private data analysis by non-experts. More recently, work by Johnson *et al* [7] has been adopted by Uber to implement differential privacy for internal data analysis[6]. The application of differential privacy to healthcare data has been reviewed by Dankar and El Emam note [1], but there don't appear to be any implementations for the OMOP CDM.

[6]https://github.com/uber/sql-differential-privacy

```
CREATE TEMPORARY TABLE top_drugs AS
SELECT
  c.concept_id,
  count(*) AS total
FROM
  drug_exposure de,
  concept c
WHERE
  de.drug_concept_id = c.concept_id AND
  c.domain_id = 'Drug'
GROUP BY
  c.concept_id
ORDER BY
  total DESC
LIMIT 8;

SELECT
  c.concept_name,
  CASE p.gender_concept_id
    WHEN 8532 THEN 'Female'
    ELSE 'Male' END AS gender,
  count(*) AS total
FROM
  person p,
  drug_exposure de,
  top_drugs td,
  concept c
WHERE
  c.concept_id = de.drug_concept_id AND
  de.drug_concept_id = td.concept_id AND
  de.person_id = p.person_id
GROUP BY
  c.concept_name,
  gender
ORDER BY
  c.concept_name,
  gender;
```

Listing 8: SQL query for the top eight most prescribed drugs broken down by gender.

7. FUTURE WORK

We would like to continue this work by exploring less simple queries like numerical aggregates and categorical data, and moving on to other tables in OMOP outside the demographics table. Our eventual goal is generate a large dataset of synthetic patients that are statistically similar to patient in the University of Washington Medical System. We want to be able to release these dataset to enable open research in the biomedical informatics community, and to give research opportunities to researchers in other fields who do not have access to clinical data.

Our future research will continue down the path of analyzing the utility of different privacy budget sizes and how those effect the quality of the data.

8. CONCLUSIONS

This project was the first step in ensuring broad public use of accurate clinical data while maintaining the privacy of the patients. Differential privacy is a promising concept that may create opportunities for meaningful and actionable discoveries with clinical data. The more we can involve people from diverse backgrounds in clinical research, the quicker we can make impactful improvements to healthcare.

9. ACKNOWLEDGMENTS

```
CREATE TEMPORARY TABLE stay_count AS
SELECT
  person_id,
  count(*)
FROM
  visit_occurrence
GROUP BY
  person_id;

SELECT CASE
  WHEN sc.count >= 0 AND sc.count <= 50
    THEN '0 - 50'
  WHEN sc.count > 50 AND sc.count <= 100
    THEN '50 - 100'
  WHEN sc.count > 100 AND sc.count <= 150
    THEN '100 - 150'
  WHEN sc.count > 150 AND sc.count <= 200
    THEN '150 - 200'
  WHEN sc.count > 200 AND sc.count <= 250
    THEN '200 - 250'
  ELSE '250+' END AS stay_length,
  COUNT(*) AS num_patients
FROM
  stay_count sc
GROUP BY
  stay_length
ORDER BY
  num_patients DESC;
```

Listing 9: SQL query for the length of stay.

Table 7: A contingency table with differentially private counts for the number of patients and the average number of days they stay in the hospital per for every visit

Stay Length (days)	Number of Patients
0 - 50	49805
50 - 100	33549
100 - 150	13210
150 - 200	2286
200 - 250	309
250+	14

The authors would like to the thank the computer science and engineer department at the University of Washington, and in particular professor Dan Suciu for his excellent class *Principles of Database Systems*, without which this work would not have been possible (or required).

```sql
SELECT
  EXTRACT('year'
    FROM date_trunc('decade',
      make_date(
        p.year_of_birth,
        p.month_of_birth,
        p.day_of_birth))
      ) AS birth_decade,
        CASE p.gender_concept_id WHEN 8532
            THEN 'Female'
        ELSE 'Male' END AS gender,
    COUNT(*)
FROM
  person p,
  condition_occurrence con_oc,
  concept con
WHERE p.person_id = con_oc.person_id
  AND con_oc.condition_concept_id
    = con.concept_id
  AND (con.concept_name LIKE '%Heart Failure%' OR
    con.concept_name LIKE '%heart failure%')
  AND con.domain_id = 'Condition'
  AND con.concept_class_id='Clinical Finding'
GROUP BY
  date_trunc('decade', make_date(p.year_of_birth,
    p.month_of_birth, p.day_of_birth)),
  p.gender_concept_id
ORDER BY
  birth_decade, gender;
```

Listing 10: SQL query for finding all patients with heart failure, grouping by gender and decade of birth.

Table 8: Table of results for the cosine similarity between the differentially private data and the original data.

Contingency Table	Cosine similarity
Heart Failure Table 3	0.99999997242
HIV Table 4	0.999832337593
Diabetes Table 5	0.999999998181
Drug Usage Table 6	0.999999814847
Length of Stay Table 7	1.0

10. REFERENCES

[1] F. K. Dankar and K. El Emam. Practicing differential privacy in health care: A review. *Transactions on Data Privacy*, 6(1):35–67, 2013.

[2] C. Dwork. Differential Privacy. *Proceedings of the International Colloquium on Automata, Languages and Programming, Part II (ICALP)*, pages 1–12, 2006.

[3] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. pages 265–284. 2006.

[5] M. Hardt, K. Ligett, and F. D. McSherry. A Simple and Practical Algorithm for Differentially Private Data Release. *Nips*, pages 1–9, 2012.

[6] G. Hripcsak, J. D. Duke, et al. Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers. *Studies in Health Technology and Informatics*, 216:574–578, 2015.

[7] N. Johnson, J. P. Near, and D. Song. Towards Practical Differential Privacy for SQL Queries. 2017.

[8] F. McSherry. Privacy integrated queries. *Communications of the ACM*, 53(9):89, 2010.

[9] F. McSherry and K. Talwar. Mechanism Design via Differential Privacy. *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, 2007.

[10] L. Sweeney. Weaving Technology and Policy Together to Maintain Confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.