实验五：交互式发布 DP 方案评估

2112852　密码科学与技术　胡佳佳

May 2024

1　实验要求

参考教材实验 5.1[1]，对交互式发布方案进行 DP 方案设计，指定隐私预算为 0.1，支持查询次数为 20 次，对 DP 发布后的结果进行评估，说明隐私保护的效果。

2　实验环境

2.1　假设背景

1. **统计查询应用**
   对一个数据集 (zoo.csv) 进行统计查询，该数据集描述了一个动物园喂食的场景，第一列中数据为动物名称，第二列中数据为动物每天消耗的胡萝卜数量。查询定义为 “每日进食超过 55 根胡萝卜的动物数量”。请设计相关的隐私保护方案，确保查询过程不泄露信息。

2. **直方图发布应用**
   在探索直方图发布的差分隐私应用场景中，以一个具体的医疗数据集为例，其数据框架建立在一个文件（例如 medicaldata.csv）之上，其中首列记录了不同的年龄段，而第二列则统计了在这些年龄段中，患有特定疾病的人数。我们打算发布的直方图，基于第一列数据作为区间分布的基础，展示了各个年龄段对应的疾病患病率。

2.2　平台

Ubuntu 18.04 虚拟机

3　理论原理

3.1　差分隐私

差分隐私概念最早由 Cynthia Dwork 等人 [2] 于 2006 年提出，区别于以往的匿名等隐私保护方案，其主要贡献是给出了对个人隐私泄露的数学定义，可以在最大化查询结果可用性的同时，保证单个用户隐私泄露不超过预先设定值。差

分隐私并不是要求保证数据集的整体性的隐私，而是对数据集中的每个个体的隐私提供保护。
基于差分隐私保护的数据发布是差分隐私研究中的核心内容，其目的是在不披露任何个人记录的情况下向公众输出汇总信息。根据对于多次查询的响应方法不同，差分隐私的发布模型分为交互式和非交互式两种:

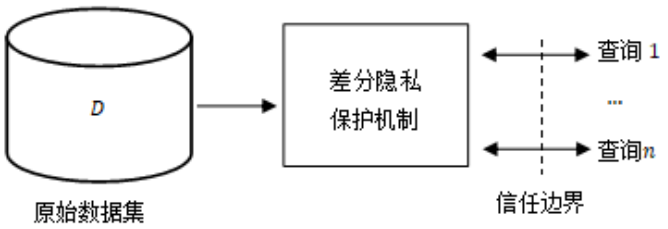

Figure 1: 交互式数据发布方案

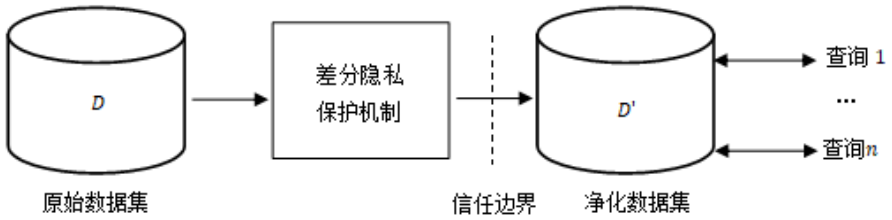

Figure 2: 非交互式数据发布方案

在非交互数据发布中，给定数据集和查询集，需通过一种数据发布机制，使其能够在满足差分隐私保护的条件下，一次性回答中的所有查询。数据管理者针对所有可能的查询，在满足差分隐私的条件下一次性发布所有查询的结果，或者发布一个原始数据集的净化版本，即带噪音的合成数据集，用户可对合成数据集自行进行所需的查询操作。非交互式数据发布方法主要集中在批查询、列联表发布、基于分组的发布方法以及净化数据集发布。
相比于交互式数据发布场景每次查询都要消耗隐私预算，非交互式只需在发布合成数据集时消耗隐私，由于差分隐私的后处理不变性，对合成数据集的后续查询任务，不会进一步泄露原始数据集的隐私。

3.2 拉普拉斯机制

我们使用 $Lap(b)$ 来表示位置参数为 0, 尺度参数为 b 的拉普拉斯分布, $Lap(x \mid b)$ 表示在尺度参数为 b 的时候输出结果为 x 的概率，即概率密度函数，表示为:

$$Lap(x \mid b) = \frac{1}{2b} exp(-\frac{|x|}{b})$$

根据差分隐私的理论，在灵敏度为 1 的时候，加入的噪声参数 b 满足为 $\frac{1}{\epsilon}$，即能满足 ϵ-差分隐私。下面，我们来形式化地定义拉普拉斯机制。

定理（拉普拉斯机制） 设函数 $f : X^n \longrightarrow R^k, \triangle f = \max_{D,D'} \|f(D) - f(D')\|_1$ 为关于函数 f 的 l_1 灵敏度。给定数据集 $D \in X^n$，则随机算法

$$M(D) = f(D) + (Y_1, ..., Y_k), Y_i \sim Lap(\triangle f/\varepsilon)$$

提供 ε-差分隐私。其中 Y_i 是独立同分布的，服从尺度参数 b 为 f/ε 的拉普拉斯分布的随机变量。

4 实验步骤

4.1 代码介绍

4.1.1 统计查询应用

本实验设计的非交互式方案不在结果上加噪音，而在数据上加噪音，产生加噪后的数据，以产生的加噪的数据集来响应查询。在这种情况下，我们对每条记录都根据设定的隐私预算产生并添加相关的拉普拉斯噪声，进而生成合成的数据集。
本部分实现置于 *testraw.c* 中。
为了对 csv 文件执行操作，首先需要设计一种能够读取 csv 文件并对其进行分析的算法 *csv_analysis*，来实现对 csv 文件的读取和其上属性的处理。该算法在第 19 行完成了对 csv 数据集的读取后，会进入一个循环体，对数据集内的数据条目进行循环处理和加噪并判断该条数据加噪后是否符合前述查询，并输出该查询的结果：

```
//循环为原始数据集内各条数据生成拉普拉斯噪音并加噪
while(original_data[i].name)
{
        //产生拉普拉斯随机数
        x = laplace_data(beta,&seed);
        //此处分别列出了每条具体添加的噪音和加噪的结果。当投入较少预算时，可能会出现负数
        printf("Added noise:%f\t%s\t%f\n",x,original_data[i].name,original_data[i].carrots+x);
        if(original_data[i].carrots+x>=55)
        {
                sum++;
        }
        i++;
}
//输出加噪后的数据集中，每日食用胡萝卜大于55的动物个数
printf("Animals which carrots cost > 55 (Under DP): %d\n",sum);
```

Figure 3: 读取 csv 文件

在主函数中，指定全局敏感度为 1，以 10 和 0.1 作为两个隐私预算，并生成基于时间的随机种子：

```c
long int seed;
int sen = 1;  //对于一个单属性的数据集，其敏感度为1
double eps[]={10,0.1}; //指定两类隐私预算，分别代表极小，极大
srand((unsigned)time( NULL )); //生成基于时间的随机种子（srand方法）
int i = 0;
```

Figure 4: 主函数参数

为了刻画信息泄露的情况，我们可以设计一个相邻数据集 $zoo_nb.csv$（去掉了“Dugong”这一项数据），来进行对比演示加入不同规模的噪音对统计结果的影响。我们利用这两个隐私预算分别基于原始数据集和相邻数据集生成加噪数据并进行前述查询，以此来对噪声的影响进行展示和比较。

```c
while(i<2)
{
        printf("Under privacy budget %f, sanitized original data with animal name and laplace noise:\n",eps[i]);
        double beta = sen / eps[i]; //拉普拉斯机制下，实际公式的算子beta为敏感度/预算
        seed = rand()%10000+10000; //随机种子产生
        csv_analysis("./zoo.csv",beta,seed); //先调用原始数据集
        printf("==================Using neighbour dataset==================\n");
        seed = rand()%10000+10000; //随机种子更新
        csv_analysis("./zoo_nb.csv",beta,seed); //再调用相邻数据集
        printf("===========================================================\n");
        i++;
}
```

Figure 5: 相邻数据集

4.1.2 直方图发布应用

直方图查询的工作机制是将整个数据集细分成若干个互不重叠的部分，每部分对应一个年龄区间，并计算每个区间内的数据点数量。鉴于这些区间之间是完全独立的，即数据集中的任何单一记录的变动仅会影响到一个特定的区间，因此，这类查询的敏感度被定为 1。基于这一特点，在每一区间的查询结果上添加根据隐私预算采样的噪声，就能够确保满足差分隐私的标准。通过这种方法，既保护了数据的隐私安全，又能够在一定程度上准确地反映出不同年龄段的疾病分布情况，这对于公共健康研究和政策制定具有重要的参考价值。
在本例中，数据集 $medicaldata.csv$ 和其相邻数据集 $md\_nb.csv$，描述了 $md\_nb.csv$ 是在 $medicaldata.csv$ 的基础上，将其中 “30-40” 区间的统计值-1 而产生的，模拟了一个场景：即一个患者决定退出医疗数据共享计划。
本部分的代码实现位于 $testhist.c$。首先，与数值型应用实现类似，我们定义了一个 $csv\_analysis$ 函数来实现对 csv 文件的读取和其上属性的处理。该算法在文件第 20 行完成了对 csv 数据集的读取后，会进入一个循环体，对数据集内的数据条目进行循环处理和加噪，并输出该区间数据加噪后的结果：

```
/*
函数功能：       对传入的csv文件进行处理，提取其中数据并生成拉普拉斯分布的噪音进行加噪
输入参数说明：
path            csv文件的存储位置
beta            拉普拉斯分布参数
seed        长整型指针变量， *seed 为伪随机数的种子
*/
void csv_analysis(char* path, double beta, long int seed)
{
        //读取指定路径的数据集
        FILE *original_file = fopen(path,"r+");
        struct Histobuckets * original_data = NULL;
        original_data = hb_csv_parser(original_file);
        int sum=0,i=0;
        double x = 0;
        //循环为原始数据集内各桶数据生成拉普拉斯噪音并加噪
        while(original_data[i].bucket)
        {
                //产生拉普拉斯随机数
                x = laplace_data(beta,&seed);
                //此处分别列出了每条具体添加的噪音和加噪的结果。当投入较少预算时，可能会出现负数
                printf("Added noise:%f\t%s\t%f\n",x,original_data[i].bucket,original_data[i].count+x);
                i++;
        }
}
```

Figure 6: 读取 csv 文件

在主函数中，我们仍然指定全局敏感度为 10 和 0.1 两个隐私预算，并生成基于时间的随机种子：

```
long int seed;
int sen = 1;  //对于一个单属性的数据集，其敏感度为1
double x;
srand((unsigned)time( NULL )); //生成基于时间的随机种子（srand方法）
double eps[]={10,0.1};
int i=0;
```

Figure 7: 主函数参数

此后，我们利用这两个隐私预算分别基于原始数据集和相邻数据集来进行加噪的直方图发布，以此来对噪声的影响进行展示和比较。

```
while(i<2)
{
        printf("Under privacy budget %f, sanitized original bucket with laplace noise:\n",eps[i]);
        double beta = sen / eps[i]; //拉普拉斯机制下，实际公式的算子beta为敏感度/预算
        seed = rand()%10000+10000; //随机种子产生
        csv_analysis("./medicaldata.csv",beta,seed); //先调用原始数据集
printf("=================Using neighbour dataset=================\n");
seed = rand()%10000+10000; //随机种子更新
csv_analysis("./md_nb.csv",beta,seed); //再调用相邻数据集
        printf("=========================================================\n");
        i++;
}
```

Figure 8: 相邻数据集

4.2 运行基础

1. 使用 VMWare Workstation 建立一台 Ubuntu18.04 虚拟机。
2. 打开 terminal，安装必要的解释器软件，此处直接使用 ubuntu 的 build-essential 来安装 gcc 和相关依赖库：

```
sudo apt-get install build-essential
```

3. 在准备进行实验的文件夹内打开终端，解压提供的 test.tar.gz 文件：

```
tar -xzvf ./test.tar.gz
```

4. 进入实验文件夹，使用 make 指令完成编译
5. 运行./testraw 和./testhist，观察运行结果

4.3 数据分析

./testraw 和./testhist 程序的运行结果都分别对预算 10 和预算 0.1 的情况进行输出。结果如下：

4.3.1 统计查询应用

在预算为 10 的情况下，

- 加噪声的前后对比：

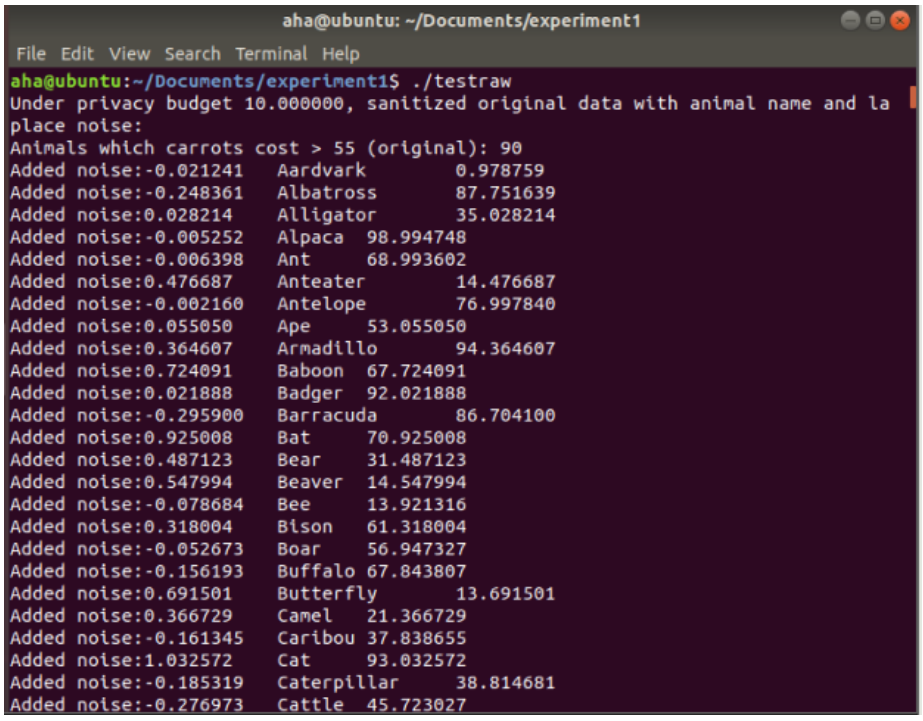

Figure 9: 预算 10 的原始数据

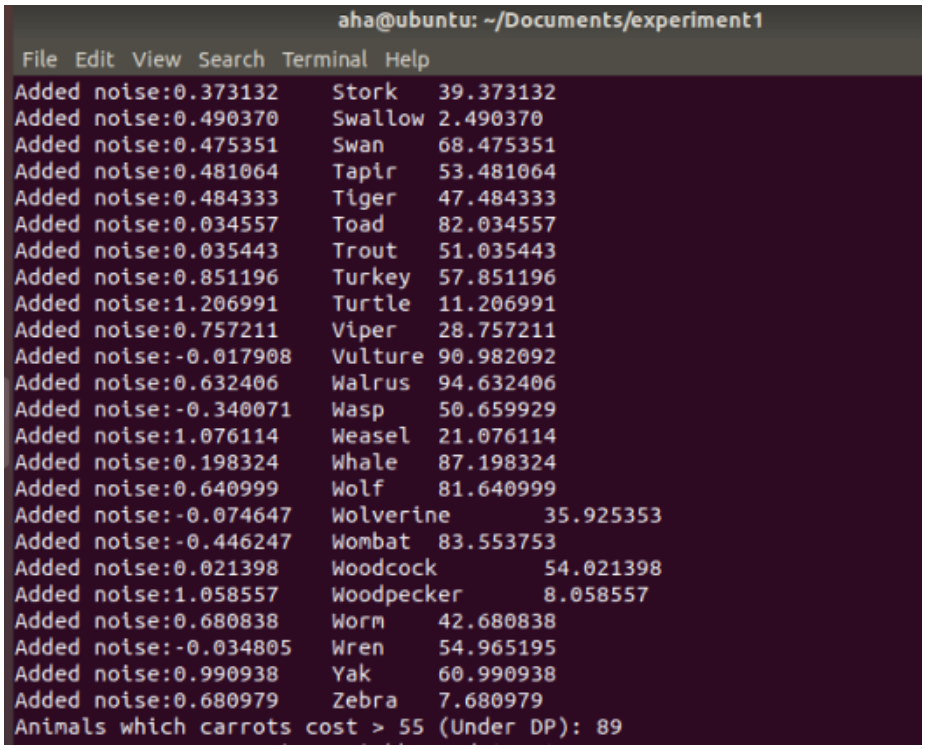

Figure 10: 预算 10 的加噪数据

可以看到，在投入较大的隐私预算的情形下，添加的噪音均小于 1 或略大于 1。对于特定查询 “每日进食大于 55 根胡萝卜的动物个数”，在该预算下，加噪前和加噪后的响应基本一致，数据可用性好。

- 相邻数据集加噪声前后对比：

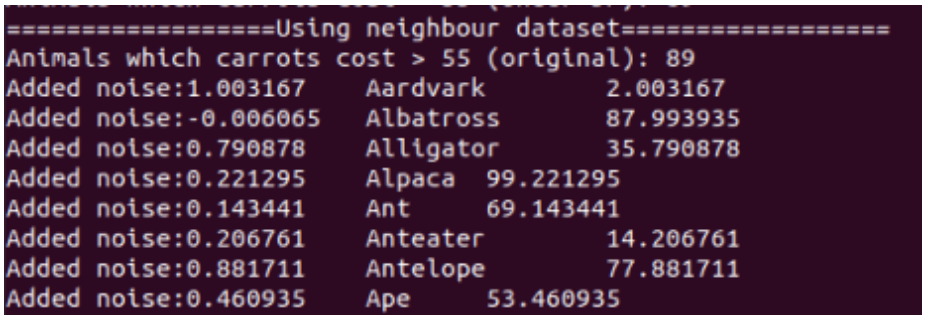

Figure 11: 预算 10 的原始相邻数据

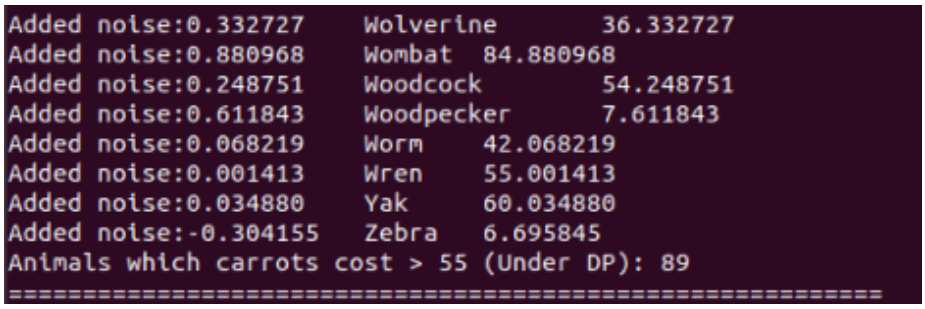

Figure 12: 预算 10 的加噪相邻数据

观察标记后对相邻数据集的处理情况，我们可以发现，加噪后数据集对该查询的响应仍与数据集的变化一致，均为 89，体现出了“Dugong”离开数据集造成的差异，不能有效抵御对该查询的差分攻击。

在预算为 0.1 的情况下，

- 加噪声的前后对比：

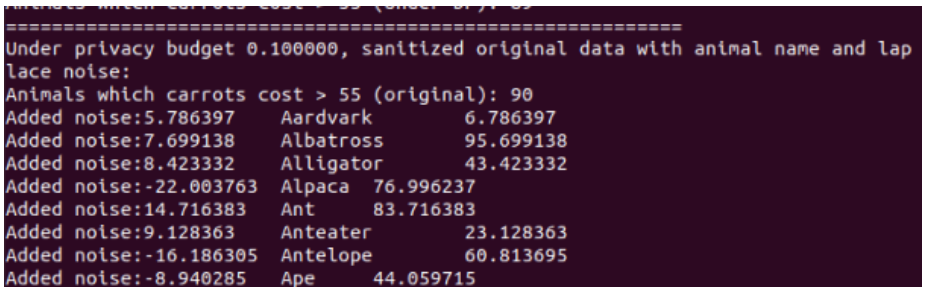

Figure 13: 预算 0.1 的原始数据

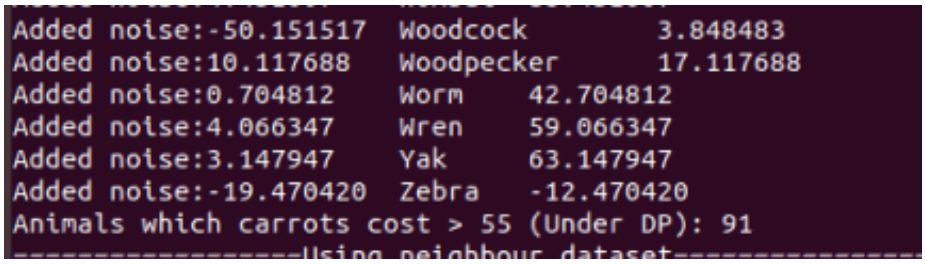

Figure 14: 预算 0.1 的加噪数据

在该预算下，产生的拉普拉斯噪音也增大了，这使得加噪后的查询结果也受到了影响。

- 相邻数据集加噪声前后对比：

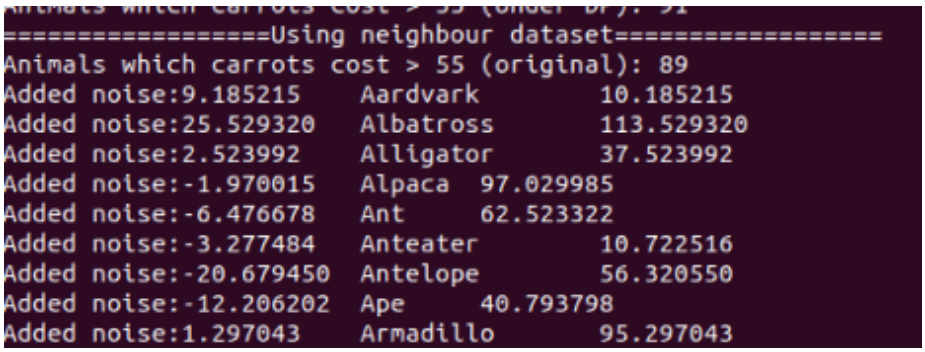

Figure 15: 预算 0.1 的原始相邻数据

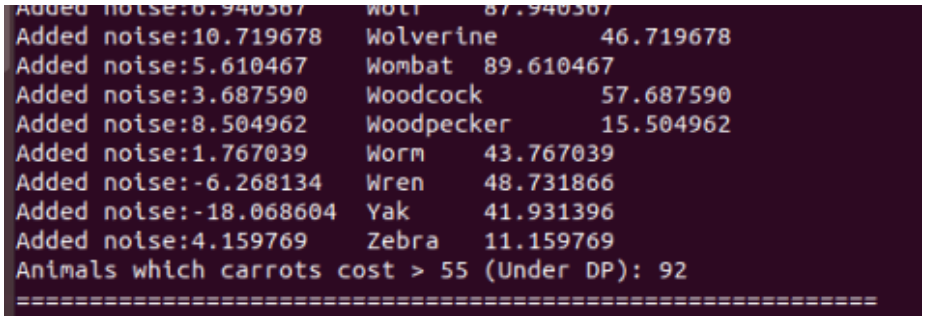

Figure 16: 预算 0.1 的加噪相邻数据

> 观察对相邻数据集进行加噪的结果，可以发现，虽然相邻数据集的直接查询结果受到了“Dugong”项移除的影响，但加噪后的相邻数据集查询结果与加噪前相比变化巨大，不再能反映出“Dugong”项移除的影响。

结合不同的预算情况，我们发现：投入较少的隐私预算时，虽然数据的可用性降低了，但是能够更好地抵御差分攻击的影响。

4.3.2 直方图发布应用

testhist 程序提供了另一种差分隐私发布方法的演示，即差分隐私的直方图发布。在该发布方式下，加噪的对象不再是数据本身，而是对数据进行分桶统计后的计数值进行加噪。
在预算为 10 和 0.1 的情况下，相邻数据集加噪声前后对比如下：

```
File  Edit  View  Search  Terminal  Help
aha@ubuntu:~/Documents/experiment1$ ./testhist
Under privacy budget 10.000000, sanitized original bucket with laplace noise:
Added noise:0.336568    20-30   405.336568
Added noise:0.570109    30-40   436.570109
Added noise:-0.180186   40-50   420.819814
Added noise:0.010522    50-60   457.010522
Added noise:0.932361    60-70   463.932361
=================Using neighbour dataset=================
Added noise:0.693801    20-30   405.693801
Added noise:0.765949    30-40   435.765949
Added noise:-0.041352   40-50   420.958648
Added noise:-0.064919   50-60   456.935081
Added noise:1.085282    60-70   464.085282
=========================================================
Under privacy budget 0.100000, sanitized original bucket with laplace noise:
Added noise:1.294694    20-30   406.294694
Added noise:6.323037    30-40   442.323037
Added noise:-2.883256   40-50   418.116744
Added noise:4.997874    50-60   461.997874
Added noise:2.827473    60-70   465.827473
=================Using neighbour dataset=================
Added noise:-14.821017  20-30   390.178983
Added noise:-36.780416  30-40   398.219584
Added noise:13.729404   40-50   434.729404
Added noise:-10.105315  50-60   446.894685
Added noise:6.735445    60-70   469.735445
=========================================================
```

Figure 17: 差分隐私的直方图分布

当隐私预算为 10 时，由于加入噪音量级较小，相邻数据集的变化仍能被体现。使用预算为 10 的差分隐私算法生成的数据和直方图如下:

分区	原始	加噪	加噪+相邻
20	405	405	405
30	436	436	436
40	421	421	421
50	457	457	457
60	463	463	463

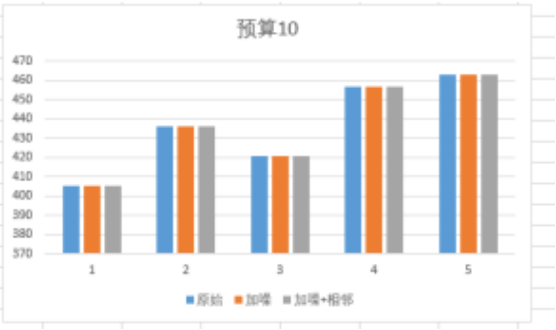

Figure 18: 预算 10 的直方图分布比较

当隐私预算为 0.1，由于噪音规模的提高，在相邻数据集的变化影响下，查询结果不减反增。即，虽然数据可用性变差，但能保护实际数据的变化不被攻击者获取，可抵御差分攻击。使用预算 0.1 的差分隐私算法生成的数据和直方图如下:

分区	原始	加噪	加噪+相邻
20	405	407	410
30	436	437	442
40	421	405	404
50	457	451	464
60	463	470	475

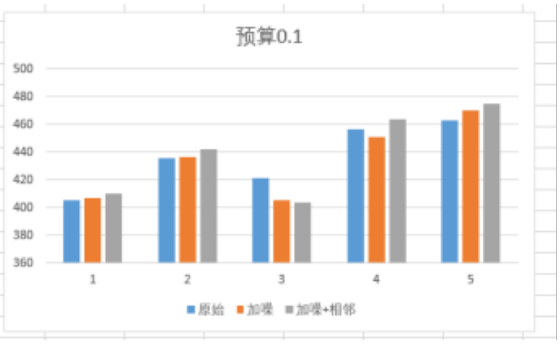

Figure 19: 预算 0.1 的直方图分布比较

5 实验总结

本次实验我们基于拉普拉斯机制设计了抗差分攻击的统机查询和直方图发布两种非交互式方案，并进行实验对不同的隐私预算进行了数据分析。我们发现：对于较高的隐私预算，具有较好的数据可用性，但在抗差分攻击方面效果不佳；而对于较低的隐私预算，则在可用性方面有所欠缺，但提供了很好的差分隐私。基于此，在实际应用中，我们需要在数据可用性与抗差分攻击性之间进行取舍，结合实际需要选择合适的隐私预算。

References

[1] 刘哲理，“数据安全课程实验教材”，2024，第六章

[2] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, “Calibrating noise to sensitivity in private data analysis,” in Theory of Cryptography: Third Theory of Cryptography Conference, 2006, pp. 265-284.