《交互式发布DP方案》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

交互式发布DP方案

**实验要求：**

根据所给的代码编译出简单的差分隐私拉普拉斯机制噪音产生和加噪程序和直方图加噪发布程序，使用给定的数据集，通过变换输入的隐私预算来观察不同隐私预算下的噪音规模和对数据的影响。

**实验过程：**

1. 差分隐私

差分隐私是Dwork在2006年首次提出的一种隐私定义，函数的输出结果对数据集中任何特定记录都不敏感。

假设对于一个考试成绩数据集D，通过查询操作得知有x个同学不及格，现加入一条新纪录得到新数据集D′，通过查询得知有x+1个同学不及格，便可推理出新加入的同学成绩不及格，如此一来，攻击者便通过这样的手段推理出了一些知识。

应对上述攻击，差分隐私通过往查询结果f(D),f(D′)中加入随机噪声r，最终得到查询结果M(D)=f(D)+r,M(D′)=f(D′)+r，使得D与D′经过同一查询后的结果并非确定的具体值，而是服从两个很接近的概率分布，这样攻击者无法辨别查询结果来自哪一个数据集，保障了个体级别的隐私性。

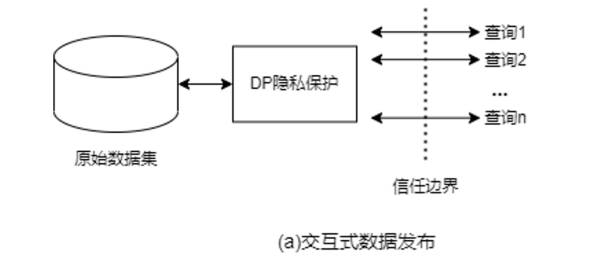
差分隐私的形式化定义如下：

定义为域中的元素，从中抽取个的集合组成数据集，其中属性的个数为维度。表示数据集D的集合，即从中抽取个组成的数据集的集合。若两个数据集和具有相同的属性结构，二者之间只有一条记录不同，则称和为相邻数据集（Neighboring Datasets）。设算法为一个随机算法，*R*为实数域。

若随机算法，对于任意子集，在任意两个相邻数据集上，满足

则称随机算法满足差分隐私。

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。



在交互式数据发布中，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，逐个回答查询集𝐹中的查询，直到耗尽全部隐私预算。即在发布前一个查询的答案之前不能发出查询。发布机制的性能通常由精确度来衡量。交互式数据发布即是要在满足一定精确度的条件下，以给定的隐私保护预算回答尽可能多的查询。

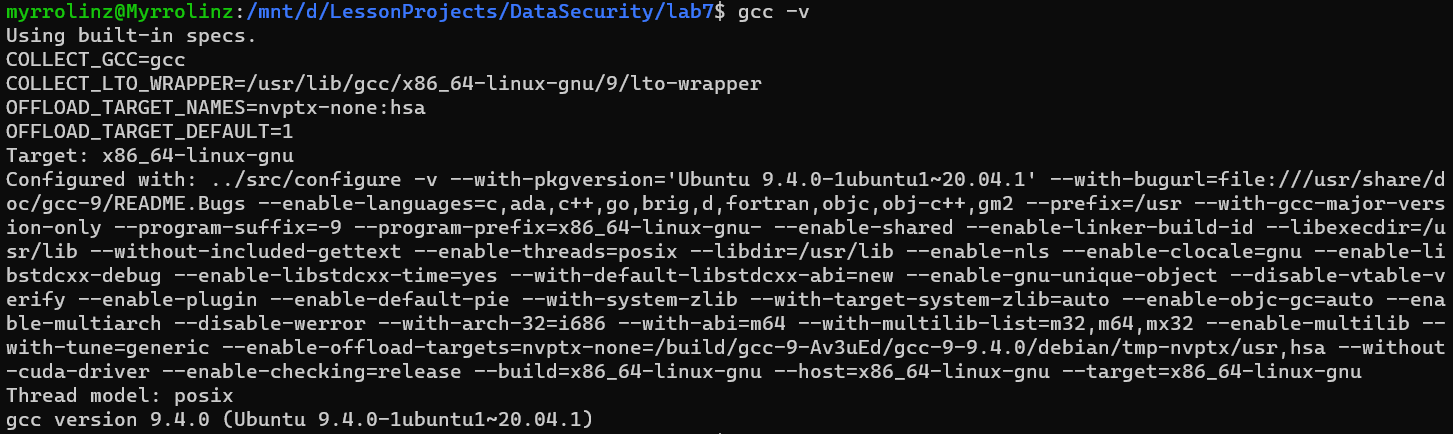
拉普拉斯机制是差分隐私中常用的随机化算法。拉普拉斯分布的特点都是在函数中值当*x*=0时，其输出最大；而在两侧，其输出呈指数型下降，且是对称的，满足这种特点的分布是对称无偏的。这个特点使得随机化算法的噪声是无偏的（即数学期望是0），进而使得噪声的均值为0，即使得水平得以大大降低，能提高数据有效性。

我们使用来表示位置参数为0、尺度参数为𝑏的拉普拉斯分布，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：

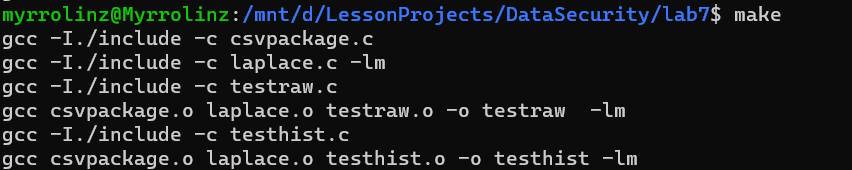
根据差分隐私的理论，在灵敏度为1的时候，加入的噪声参数满足为，即能满足-差分隐私。

1. 安装依赖库与编译

打开ubuntu（此处为wsl2），安装build-essential。使用指令gcc -v查看版本：



解压test.tar.ga，并执行make指令：



1. 数据集说明

**zoo.csv**：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”。

**zoo\_nb.csv**：相邻数据集（在zoo.csv的基础上去掉了“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。

**medicaldata.csv**：其第一列数据为一特定年龄区间，第二列数据为该年龄区间患有某种疾病的人数。要发布的直方图即是以第一列数据为直方图的桶（Bucket）发布的统计数据。

**md\_nb.csv**：相邻数据集，是在medicaldata.csv的基础上将其中“30-40”区间的统计值-1而产生的，模拟原始医疗数据集内一名患者退出数据共享。

1. 解读重要函数

**Csvpackage.c**： 这个函数主要完成csv的文件的读取，对原始数据做一些变换，或将原始数据存储为其他结构体，不是我们讨论的重点，这里我们略过此部分。

**Laplace.c：**

double uniform\_data(double a, double b,long int \* seed)

{

    double t;

    \*seed = 2045.0 \* (\*seed) + 1; // 2045.0 = 2^11 + 1

    \*seed = \*seed - (\*seed / 1048576) \* 1048576; // 1048576 = 2^20

    t = (\*seed) / 1048576.0;

    t = a + (b - a) \* t; // 生成[a,b]区间上的均匀分布随机数

    return t;

}

Uniform\_data主要是利用混合同余法产生（a,b）区间上均匀分布的随机数。其中，a为给定区间的下限，b为给定区间的上限，\*seed是伪随机数种子。线性同余法的公式为.此处我们设定。其中，2045和1048576是线性同余法中使用的常数，选择它们是为了产生高质量的伪随机数序列。具体来说，2045是一个较大的质数，它可以确保生成的随机数序列具有较长的周期，即生成的随机数序列不会很快地重复。

double laplace\_data(double beta, long int \* seed)

{

    double u1,u2, x;

    u1 = uniform\_data(0.0, 1.0, seed); // 生成[0,1]区间上的均匀分布随机数

    u2 = uniform\_data(0.0, 1.0, seed); // 生成[0,1]区间上的均匀分布随机数

    if (u1 < 0.5)   // 生成拉普拉斯分布随机数

    {

        x = beta \* (log(2\*u1)+u2);

    }

    else{

        x = u2 - (beta \* log(2\*(1-u1)));

    }

    return x;

}

Lapace\_data主要用于求解laplace分布概率累积的反函数，并利用该反函数产生laplace分布的随机数。其数学过程如下：

1. 首先，产生均匀分布的随机数和,即

2. 计算

**Testraw.c:**

第一个函数csv\_analysis，它主要是传入的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。其中有这样一段代码：

while(original\_data[i].name)  //循环为原始数据集内各条数据去除标识（动物名）、生成拉普拉斯噪音并加噪

    {

        x = laplace\_data(beta,&seed); //产生拉普拉斯随机数

        printf("Added noise:%f\t%s %d\t%f\n",x,"Animal",i+1,original\_data[i].carrots+x); //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

        if(original\_data[i].carrots+x>=55)

        {

            sum++;

        }

        i++;

    }

    printf("Animals which carrots cost > 55 (Under DP): %d\n",sum); //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

在这个循环中，对数据的每一个条目进行加噪，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。

在main()函数中，默认敏感度int sen = 1;设置beta = 0.

**Testhist.c:**

Csv\_analysis函数与testraw.c大致相同，但是里面的循环体不同：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。

    while(original\_data[i].bucket)  //循环为原始数据集内各桶数据生成拉普拉斯噪音并加噪

    {

        x = laplace\_data(beta,&seed); //产生拉普拉斯随机数

        printf("Added noise:%f\t%s\t%f\n",x,original\_data[i].bucket,original\_data[i].count+x); //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

        i++;

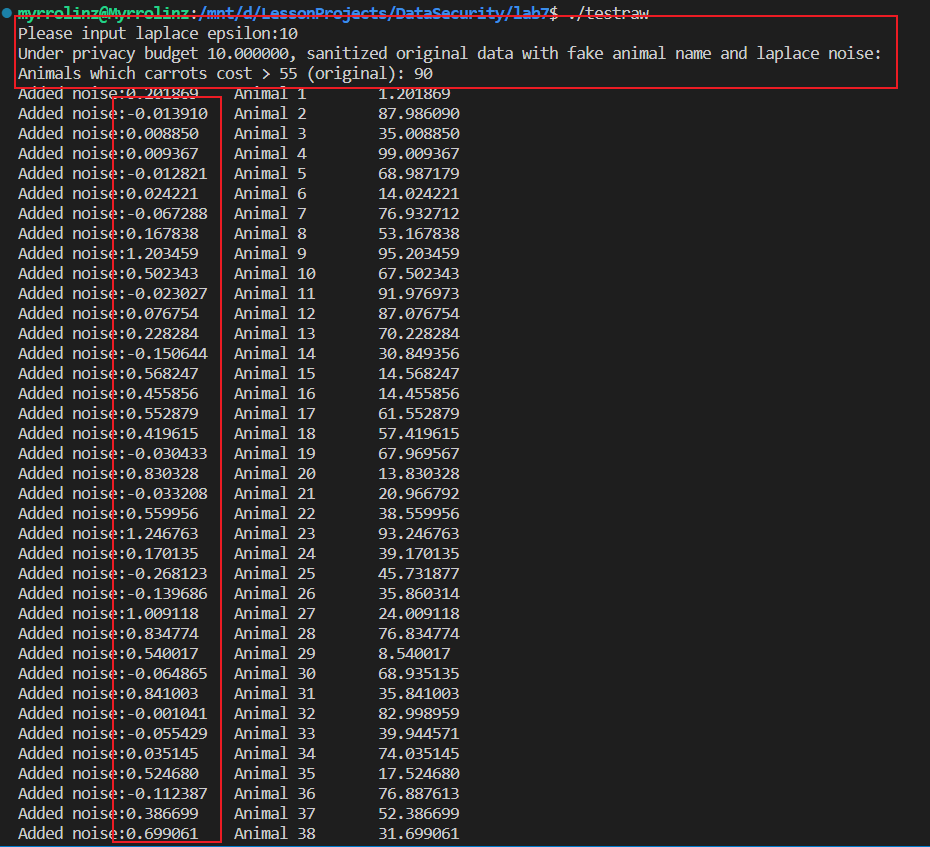
    }

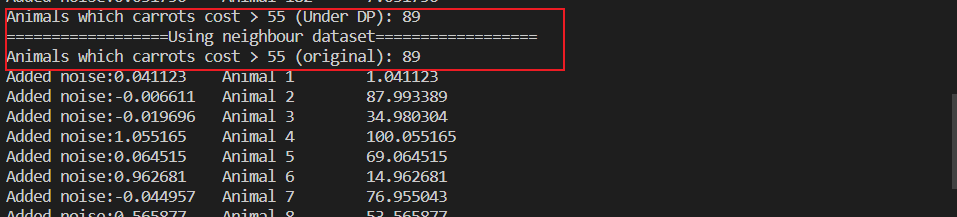
其余与testraw函数大致相同。

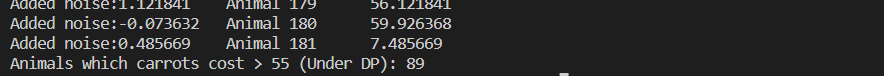
1. 运行testraw

在这里将隐私预算指定为10，运行查看结果：

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一致。原始情况响应为90，加噪后响应为89，相差很小，数据可用性更好：

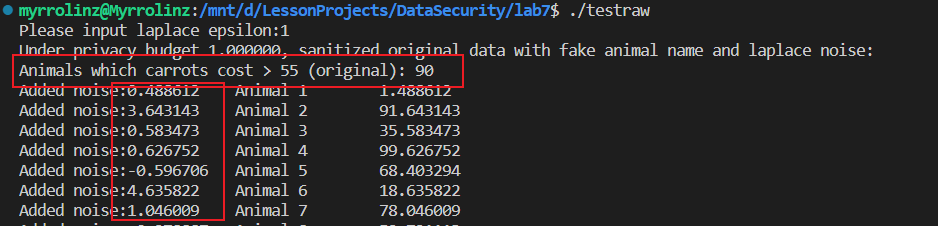




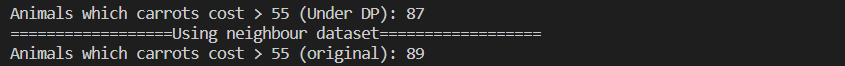


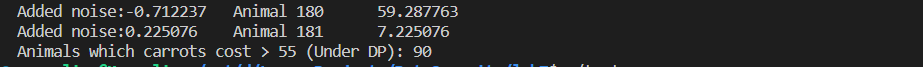
但是，观察标记后对相邻数据集的处理情况，我们可以发现，加噪后数据集对该查询的响应为89，仍与数据集的变化一致，不能有效抵御对该查询的差分攻击。

再次尝试，投入更少的隐私预算，如1：



我们可以看到，产生的拉普拉斯噪音更大的，这将对加噪后的查询结果产生影响。

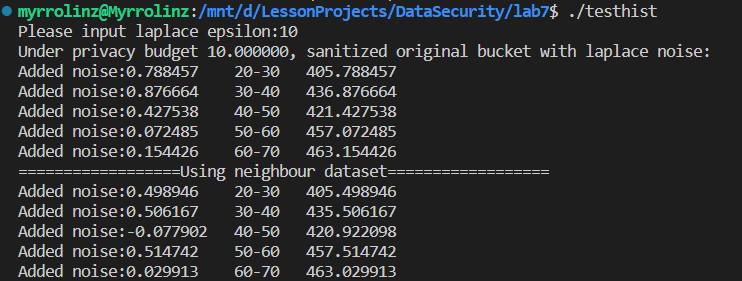




观察相邻数据集，原始查询是89，相对于90少1（因为移除了Dugeng项），但是加噪后的查询结果为90，与原始数据集相同，与数据集变化方向不同，不再能反映出“Dugong”项移除的影响。这就意味着，投入较少的隐私预算时，虽然数据的可用性降低了，但是能够更好地抵御差分攻击的影响。

6. 差分隐私的直方图发布

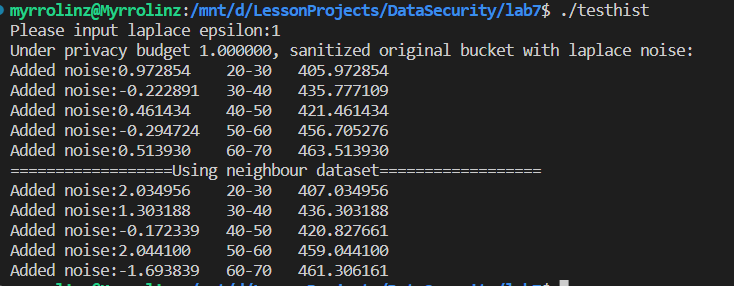
执行Testhist程序，使隐私预算为10，得到以下结果：



可以看到，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。在Excel将其制作为图表，展示如下：

直观地说，加噪后的数据和原始数据并没有太大的差别。

再次尝试将隐私预算设置为1：



可以看到，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。

7. 隐私预算为0.1，查询次数20轮

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **噪声绝对值的数据分布** | | | | **多次查询添加噪声的平均值落在危险区间内的概率** | | | |
| 90% | 95% | 99% | 99.9% | 100 | 1000 | 10000 | 10000 |
| **1** | 2.29 | 2.98 | 4.50 | 6.43 | 100.00 | 100.00 | 100.00 | 100.00 |
| **0.1** | 23.24 | 29.99 | 45.51 | 66.56 | 25.59 | 73.75 | 99.99 | 100.00 |
| **0.01** | 227.97 | 296.22 | 463.48 | 677.26 | 2.72 | 9.12 | 27.85 | 73.70 |

事实上，隐私预算刻画了一类查询任务的总体允许的隐私泄露程度。如果仅仅将作为生成拉普拉斯噪音的参数的话，如上表所示，多次查询很容易就实现隐私信息的推断。

要考虑保护多次查询的话，需要为每次查询进行预算分配：假定隐私预算为，允许的查询次数为*k*，则每次查询分配的预算为，这样才能达到-差分隐私的目标。

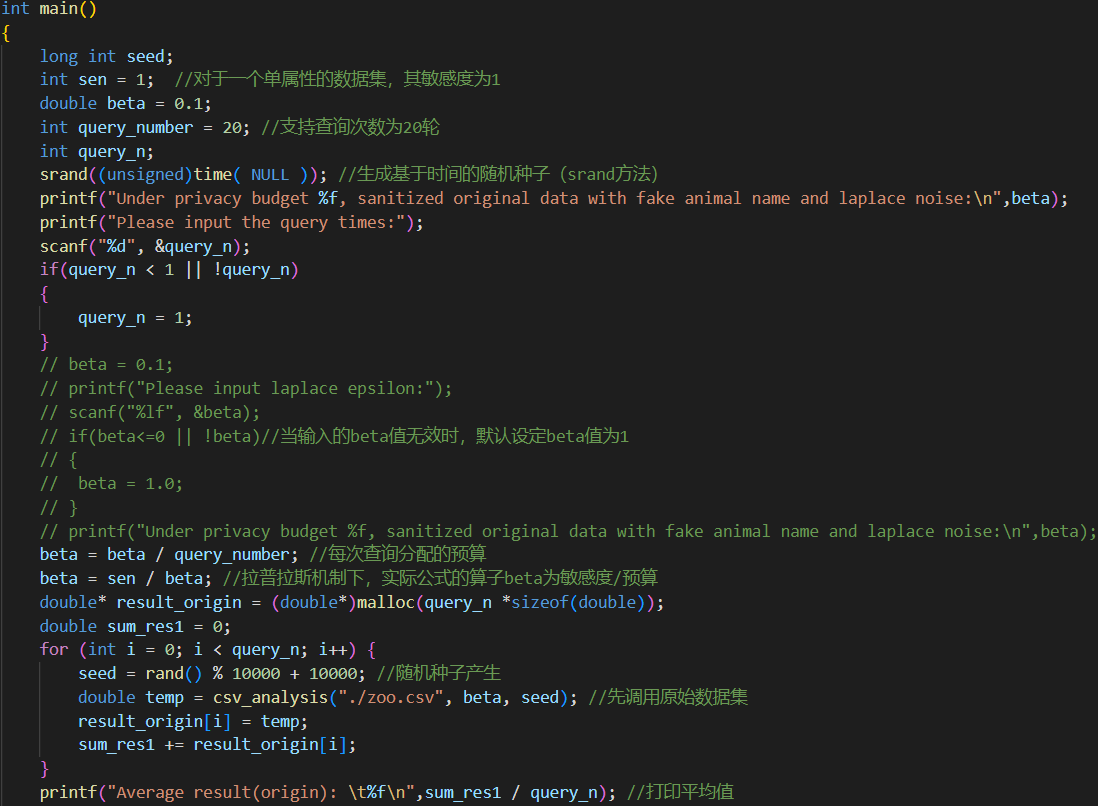
因此，对于统计查询而言，如果在查询结果上进行反馈，则需要定义所能支持的次数，进而按上述方式对每次查询进行预算的分配。换句话说，这种添加噪音的方式，会使得每次查询都会消耗一定的隐私预算，直到隐私预算都被消耗干净，就再也不能起到保护的作用。

为了实现隐私预算为0.1，查询次数20轮，修改csv\_analysis函数，代码实现如下：



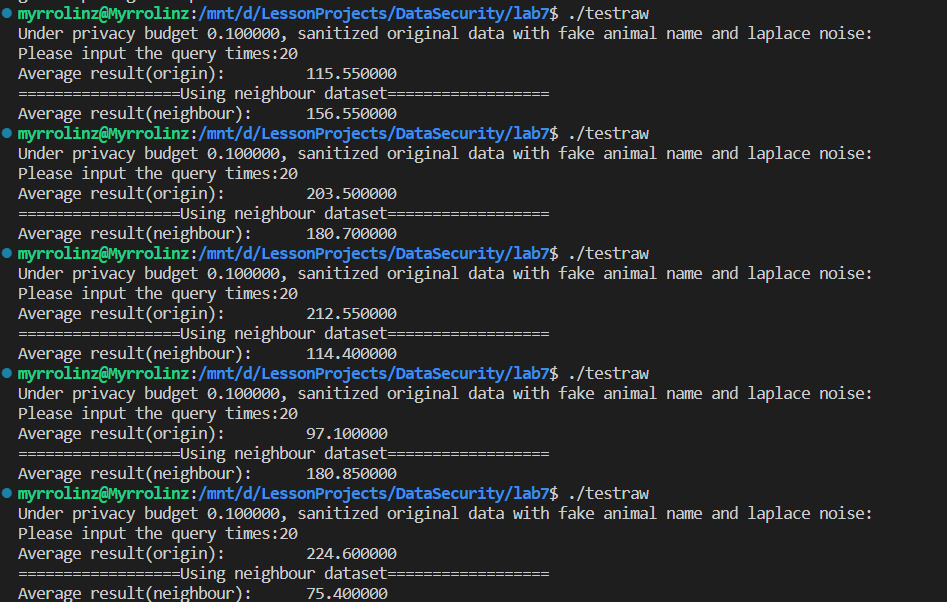
每次查询都会消耗一定的隐私预算，因此修改main函数的实现如下。设置beta为定值0.1，变量query\_number为查询次数20，每次查询分配的预算为beta/query\_number，变量query\_n为用户输入的查询次数，定义一个大小为 n\_query 的数组来记录每一次的值，用于后续求平均，模拟重复推测攻击过程，评估隐私保护的效果。

然后需要修改main()函数，以支持20次隐私查询：





实验效果如下所示：



可以看到，在隐私预算为0.1时，原始查询较隐私预算为1时偏差较大，在相邻数据集上也相较于原始查询表现出了较大偏差，难以探寻数据集变化的踪迹。隐私保护效果较好。

**心得体会：**

本次实验感觉较前几次实验复杂了些，但也更有意思。实验过程并不难，直观地让我感受到了隐私预算变化带来的查询结果的变化，加深了我对DP方案的理解。

通过本次实验，我了解了差分攻击和差分隐私的概念，实现了拉普拉斯噪声生成过程，掌握了交互式数据发布和非交互式数据发布的区别。

作为扩展，我还了解到了其它差分隐私的随机化算法，如指数机制、高斯机制等；在这些机制中，噪声的发现取决于算法的敏感度。

通过本次实验，我受益良多。