

南开大学

数据安全 课程实验报告

交互式发布 *DP* 方案评估



学院 网络空间安全学院

专业 信息安全

姓名 齐明杰

学号 2113997

2024 年 4 月 27 日

目 录

1	实验目的	3
2	实验原理	3
2.1	隐私预算的设定与管理	3
2.2	拉普拉斯噪声的计算与添加	3
2.3	隐私保护的效果评估	3
3	实验过程	4
3.1	实验环境	4
3.2	非交互式方案	6
3.3	交互式 DP 方案	7
3.4	运行结果	9
4	实验心得	12

图表

表 2.3.1: 交互式发布的概率表	4
图 3.1.2: 实验代码	4
图 3.1.3: gcc 安装状态	5
图 3.1.4: 编译代码	5
图 3.1.5: 修改代码	5
图 3.1.6: 重新 make 编译	5
图 3.2.7: $\epsilon = 10$ 结果	6
图 3.2.8: $\epsilon = 10$ 加噪后	6
图 3.2.9: $\epsilon = 10$ 相邻数据集	6
图 3.2.10: $\epsilon = 10$ 相邻数据集加噪后	6
图 3.2.11: $\epsilon = 0.1$ 结果	7
图 3.2.12: $\epsilon = 0.1$ 加噪后	7
图 3.2.13: $\epsilon = 0.1$ 相邻数据集加噪后	7
图 3.3.14: 修改 csv_analyse 函数	8
图 3.4.15: $\epsilon = 10$	10
图 3.4.16: $\epsilon = 10$ 邻近集	10
图 3.4.17: $\epsilon = 10$ 邻近集平均值	10
图 3.4.18: $\epsilon = 0.1$ 平均值	11
图 3.4.19: $\epsilon = 0.1$ 在邻近集的平均值	11
图 3.4.20: $\epsilon = 0.1$ 平均值	11
图 3.4.21: $\epsilon = 0.1$ 在邻近集的平均值	11

1 实验目的

交互式发布 DP 方案评估

参考教材实验 5.1，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 实验原理

在本实验中，我们将通过设计一个基于拉普拉斯机制的差分隐私 (DP) 方案，对数据进行隐私保护处理。实验目的是在保证数据隐私的前提下，对动物园日常统计数据（如每天消耗胡萝卜数量）进行安全查询。我们选择的是交互式发布方案，这意味着在每次查询响应时，系统会动态地向结果中添加噪声，以防止通过多次查询推断出数据集中的真实信息。

差分隐私技术的核心在于，通过在查询结果中添加一定量的随机噪声，来防止攻击者即使拥有除某一数据外的所有其他数据，也无法确定该数据是否存在于数据集中。本实验采用的噪声添加机制是拉普拉斯机制，它依据设定的隐私预算 (ϵ) 来调整噪声的量级。。

2.1 隐私预算的设置与管理

在本实验中，整体隐私预算设定为 0.1，意味着极高的隐私保护级别。根据差分隐私的理论，较小的 ϵ 值（隐私预算）将导致添加更大的噪声，以保护数据的隐私性。为了控制整体隐私泄露，假设允许用户进行 20 次查询，每次查询将消耗总隐私预算的一部分，即每次查询的隐私预算为 $0.1/20$ 。

2.2 拉普拉斯噪声的计算与添加

拉普拉斯噪声的计算公式为：

$$\text{Laplace noise} = \frac{\Delta f}{\epsilon}$$

其中， Δf 是查询函数的敏感度，本实验中设为 1（单个记录的最大影响）。因此，每次查询添加的噪声量由敏感度与每次查询的隐私预算决定。

2.3 隐私保护的效果评估

通过实验，我们将观察在不同隐私预算下，噪声对查询结果的影响。在隐私预算极低（如 0.1）的情况下，预期噪声较大，查询结果的准确性会受到较大影响，但隐私保护的效果更为显著。这种权衡是评估差分隐私实用性的关键部分。

重复攻击是针对差分隐私的攻击方式。因为拉普拉斯机制添加噪声的特点是无偏估计，

多次查询后均值为 0，如果攻击者向数据库请求重复执行同一个查询语句，将结果求和平均，就有极大的概率获得真实结果。

ϵ	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

表 2.3.1: 交互式发布的概率表

3 实验过程

3.1 实验环境

先把代码拷入 Ubuntu 22 x64 虚拟机：

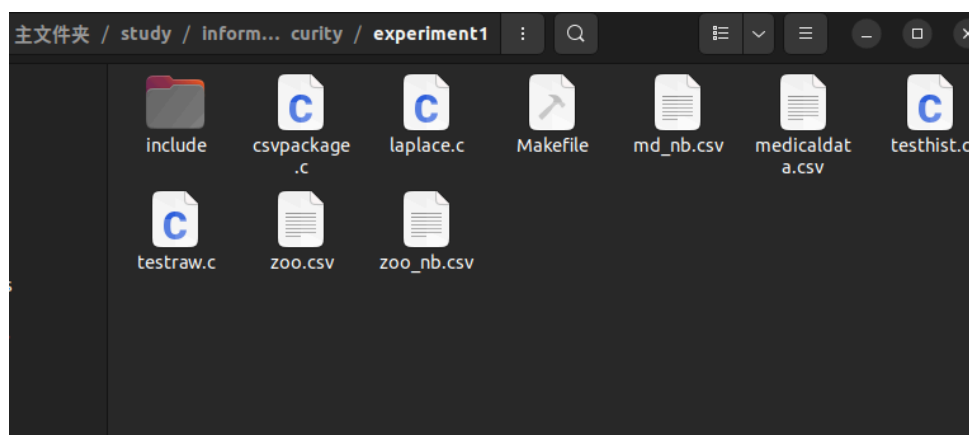


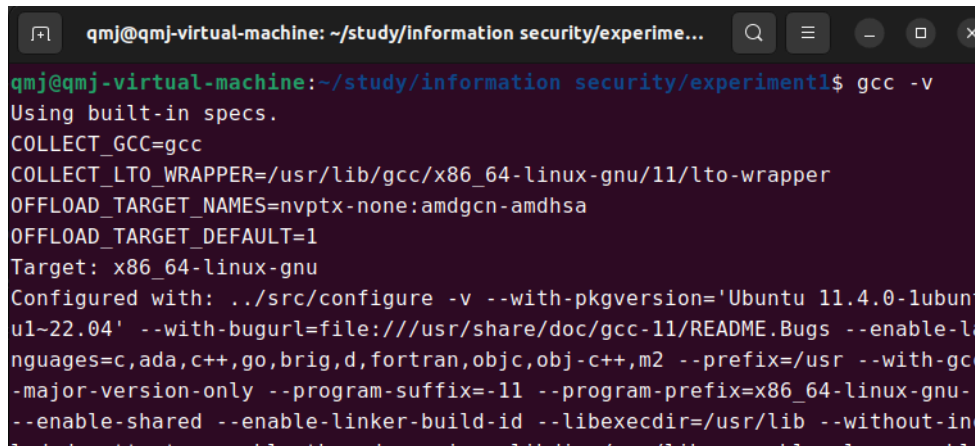
图 3.1.2: 实验代码

项目的结构如下所示：

```

└─ experiment1
    │
    └─ include
        │
        └─ csvpackage.h 为 csvpackage 函数族提供函数定义
            └─ laplace.h 为 laplace 函数族提供函数定义
    └─ csvpackage.c 实验代码的 csv 读取和预统计函数实现
    └─ laplace.c 为实验代码生成拉普拉斯分布的随机数
    └─ Makefile 用于编译使用的编译规则文件
    └─ md_nb.csv 将 medicaldata.csv 中“30-40”区间的统计值-1
    └─ medicaldata.csv 本次实验直方图发布部分的样例数据集
    └─ testhist.c 数据集直方图发布加噪实验部分的主函数
    └─ testraw.c 数据集直接加噪实验部分的主函数
  
```

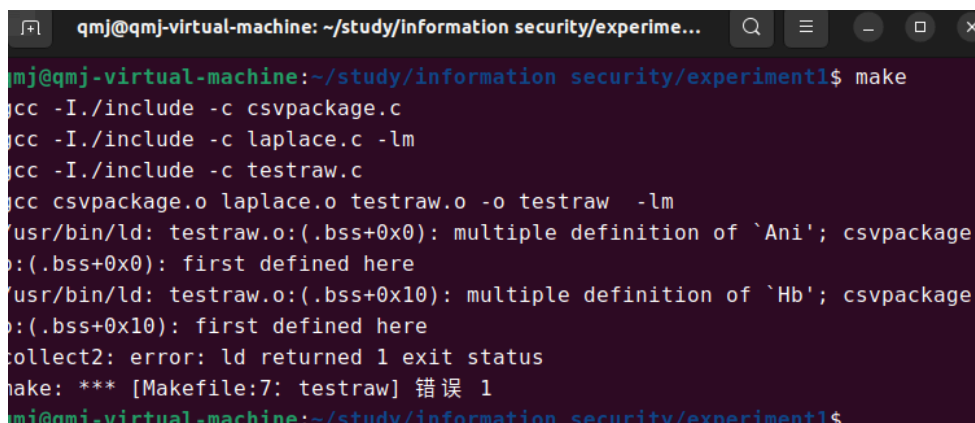
- └ zoo.csv 本次实验使用的样例数据集
 - └ zoo_nb.csv 将 zoo.csv 中“Dugeng”项去除得到的相邻数据集
- 然后确认 gcc 安装状态: `gcc -v`



```
qmq@qmqj-virtual-machine: ~/study/information security/experiment1$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgc- amdhsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 11.4.0-1ubuntu1-22.04' --with-bugurl=file:///usr/share/doc/gcc-11/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,m2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-11 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-headers --enable-threads=posix --enable-objc-gate=0 --enable-objc-gate=0 --enable-objc-gate=0
```

图 3.1.3: gcc 安装状态

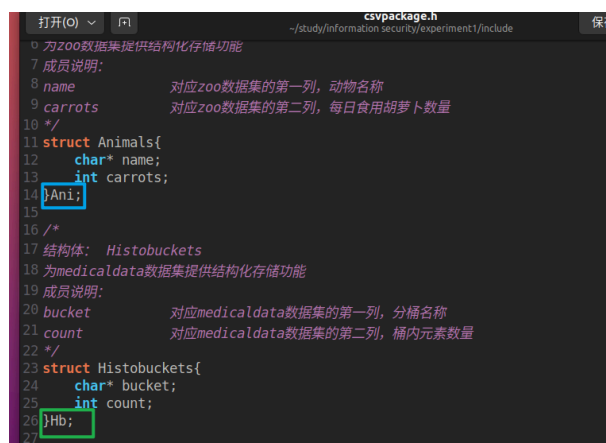
使用 makefile 编译代码:



```
qmq@qmqj-virtual-machine: ~/study/information security/experiment1$ make
gcc -I./include -c csvpackage.c
gcc -I./include -c laplace.c -lm
gcc -I./include -c testraw.c
gcc csvpackage.o laplace.o testraw.o -o testraw -lm
/usr/bin/ld: testraw.o(.bss+0x0): multiple definition of `Ani'; csvpackage.o(.bss+0x0): first defined here
/usr/bin/ld: testraw.o(.bss+0x10): multiple definition of `Hb'; csvpackage.o(.bss+0x10): first defined here
collect2: error: ld returned 1 exit status
make: *** [Makefile:7: testraw] 错误 1
qmq@qmqj-virtual-machine: ~/study/information security/experiment1$
```

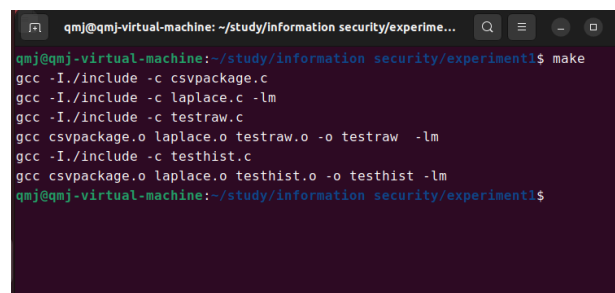
图 3.1.4: 编译代码

发现报错了, 实验所提供的参考代码在 ubuntu22 以及最新版本的 gcc 的环境下并不能够顺利编译通过, 经检查发现这是由于重定义问题, 我们删除 csvpackage.h 中的 Ani 和 Hb 实例, 编译通过:



```
打开(O)  csvpackage.h 保存
~/study/information security/experiment1/include
/* 为 zoo 数据集提供结构化存储功能
7 成员说明:
8 name          对应 zoo 数据集的第一列, 动物名称
9 carrots       对应 zoo 数据集的第二列, 每日食用胡萝卜数量
10 */
11 struct Animals{
12     char* name;
13     int carrots;
14     Ani;
15 }
16 /*
17 结构体: Histobuckets
18 为 medicaldata 数据集提供结构化存储功能
19 成员说明:
20 bucket        对应 medicaldata 数据集的第一列, 分桶名称
21 count         对应 medicaldata 数据集的第二列, 桶内元素数量
22 */
23 struct Histobuckets{
24     char* bucket;
25     int count;
26     Hb;
27 }
```

图 3.1.5: 修改代码



```
qmq@qmqj-virtual-machine: ~/study/information security/experiment1$ make
gcc -I./include -c csvpackage.c
gcc -I./include -c laplace.c -lm
gcc -I./include -c testraw.c
gcc csvpackage.o laplace.o testraw.o -o testraw -lm
gcc -I./include -c testhist.c
gcc csvpackage.o laplace.o testhist.o -o testhist -lm
qmq@qmqj-virtual-machine: ~/study/information security/experiment1$
```

图 3.1.6: 重新 make 编译

3.2 非交互式方案

运行 testraw 程序, 观察生成的噪音和加噪后的数据与原始数据的差别。代码分别运行了预算为 10 和 0.1 时候的结果:

```
qmj@qmj-virtual-machine:~/study/information security/experiment1$ ./testraw
Under privacy budget 10.000000, sanitized original data with animal name and
laplace noise:
Animals which carrots cost > 55 (original): 90
Added noise:1.080292   Aardvark      2.080292
Added noise:0.536073   Albatross     88.536073
Added noise:-0.019401  Alligator     34.980599
Added noise:0.012241   Alpaca        99.012241
Added noise:0.005951   Ant           69.005951
Added noise:0.421120   Anteater      14.421120
Added noise:0.178006   Antelope      77.178006
Added noise:-0.033216  Ape           52.966784
```

图 3.2.7: $\epsilon = 10$ 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, 数据可用性好:

```
Added noise:0.552299   Zebra        7.552299
Animals which carrots cost > 55 (Under DP): 89
```

图 3.2.8: $\epsilon = 10$ 加噪后

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仍与数据集的变化基本一致, 均为 89, 体现出了 Dugeng 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。

```
=====Using neighbour dataset=====
Animals which carrots cost > 55 (original): 89
Added noise:0.047087   Aardvark      1.047087
```

图 3.2.9: $\epsilon = 10$ 相邻数据集

```
Added noise:-0.049211  Zebra        6.950789
Animals which carrots cost > 55 (Under DP): 89
=====
Under privacy budget 0.100000, sanitized original data with
```

图 3.2.10: $\epsilon = 10$ 相邻数据集加噪后

我们此时再输入 0.1 作为隐私预算。可以看到, 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:


```

qmqj@qmqj-virtual-machine: ~/study/Information security/experime...
Added noise:-0.049211  Zebra  6.950789
Animals which carrots cost > 55 (Under DP): 89
=====
Under privacy budget 0.100000, sanitized original data with animal name and
laplace noise:
Animals which carrots cost > 55 (original): 90
Added noise:12.866510  Aardvark  13.866510
Added noise:7.323035  Albatross  95.323035
Added noise:-8.214362  Alligator  26.785638
Added noise:0.184143  Alpaca  99.184143
Added noise:7.625348  Ant  76.625348
Added noise:14.798583  Anteater  28.798583
Added noise:1.278098  Antelope  78.278098
Added noise:-6.710095  Ape  46.288015

```

图 3.2.11: $\epsilon = 0.1$ 结果

```

Added noise:13.554052  Zebra  20.554052
Animals which carrots cost > 55 (Under DP): 94
=====Using neighbour dataset=====

```

图 3.2.12: $\epsilon = 0.1$ 加噪后

但是同时，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“Dugeng”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“Dugeng”项移除的影响：

```

Added noise:-2.398728  Zebra  4.601272
Animals which carrots cost > 55 (Under DP): 101
=====

```

图 3.2.13: $\epsilon = 0.1$ 相邻数据集加噪后

即，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。

3.3 交互式 DP 方案

在交互式发布 DP 方案中，系统会对每次外部的查询请求动态添加噪声。这意味着每次查询结果都通过引入随机噪声来模糊真实值，以此来保护数据集中个体的隐私。这种方法的关键在于如何分配隐私预算和计算所需的噪声量，以维持整体系统的隐私保护效果而不至于在多次查询后隐私泄露累积到可被利用的程度。

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 ϵ ，允许的查询次数为 k ，则每次查询分配的预算为 $\frac{\epsilon}{k}$ ，这样才能达到 ϵ -差分隐私的目标。在上述表中我们已经列出了重复隐私攻击的概率。

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，按照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，会使每次查询都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。

因此，我们需要修改一部分代码，来达到我们的目的：

首先修改 testraw.c 的 csv_analyse 函数，让其能够在每一次分析返回一个值：

```

1 double csv_analysis(char* path, double beta, long int seed)
2 {
3     FILE *original_file = fopen(path,"r+"); //读取指定路径的数据集
4     struct Animals * original_data = NULL;
5     original_data = csv_parser(original_file);
6     int sum=0,i=0;
7     double x = 0;
8     while(original_data[i].name) //循环为原始数据集内各条数据生成拉普拉斯噪音并加噪
9     {
10         x = laplace_data(beta,&seed); //产生拉普拉斯随机数
11         // printf("Added noise:%f\t%s\t%f\n",x,original_data[i].name,original_data[i].carrots+x);
12         //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
13         if(original_data[i].carrots+x>=55)
14         {
15             sum++;
16         }
17         i++;
18     }
19     printf("Animals which carrots cost > 55 (Under DP): %d\n",sum+(int)x);
20     // 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
21     return sum + x;
22 }

```

图 3.3.14: 修改 csv_analyse 函数

然后修改 testraw.c 的 main 函数，添加相关次数变量：

```

1 int main()
2 {
3     long int seed;
4     int sen = 1; //对于一个单属性的数据集，其敏感度为 1
5     double eps[]={10,0.1}; //指定两类隐私预算，分别代表极小，极大
6     srand((unsigned)time( NULL )); //生成基于时间的随机种子 (srand 方法)
7     int i = 0;
8
9     int search_times = 20;
10    printf("Input search times is %d\n", search_times);
11
12    while(i<2)
13    {
14        printf("Under privacy budget %f, sanitized original data with animal
15        name and laplace noise:\n",eps[i]);
16        double beta = sen / (eps[i] / search_times); //拉普拉斯机制下，实际公式的
17        算子 beta 为敏感度/预算
18        // seed = rand()%10000+10000; //随机种子产生
19        // csv_analysis("./zoo.csv",beta,seed); //先调用原始数据集
20    }
21 }

```



```

19     double avg_old=0;
20     for(int i=0;i<search_times;i++){
21         seed = rand()%10000+10000; //随机种子产生
22         avg_old += csv_analysis("./zoo.csv",beta,seed); //先调用原始数据集
23     }
24     printf("Avg Old Search Result: \t%f\n", avg_old/search_times);
25
26         printf("=====Using      neighbour
dataset=====\\n");
27     // seed = rand()%10000+10000; //随机种子更新
28     // csv_analysis("./zoo_nb.csv",beta,seed); //再调用相邻数据集
29
30     double avg_new=0;
31     for(int i=0;i<search_times;i++){
32         seed = rand()%10000+10000; //随机种子更新
33         avg_new += csv_analysis("./zoo_nb.csv",beta,seed); //再调用相邻数据集
34     }
35     printf("Avg New Search Result: \t%f\n", avg_new/search_times);
36
37     printf("=====\\n");
38     i++;
39 }
40 return 0;
41 }

```

这里我们设置 查询次数为 20

3.4 运行结果

重新编译上述修改好的代码，得到如下结果：

当 $\varepsilon = 10$ 时候的结果：

```

qmj@qmj-virtual-machine: ~/study/information security/experime...
qmj@qmj-virtual-machine:~/study/information security/experiment1$ ./testraw
Input search times is 20
Under privacy budget 10.000000, sanitized original data with animal name and
laplace noise:
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 87
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 87
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 86
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 92
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 90
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 93

```

图 3.4.15: $\epsilon = 10$

```

Animals which carrots cost > 55 (Under DP): 91
Avg Old Search Result: 89.921345
=====Using neighbour dataset=====
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 88
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 82
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 88
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 90
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 92
Animals which carrots cost > 55 (original): 89

```

图 3.4.16: $\epsilon = 10$ 邻近集

```

Animals which carrots cost > 55 (Under DP): 88
Avg New Search Result: 87.410236
=====

```

图 3.4.17: $\epsilon = 10$ 邻近集平均值

不难发现，此时由于预算很大，我们每个噪声都很小，因此修改不大，并且平均值也变化不大，约为 89.92 和 87.41。

我们主要观察 $\epsilon = 0.1$ 时候的结果。

当 $\epsilon = 0.1$ 时候的结果：

```

Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): -736
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): -38
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 244
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 65
Avg Old Search Result: 104.812986

```

图 3.4.18: $\epsilon = 0.1$ 平均值

```

Animals which carrots cost > 55 (Under DP): 967
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): -616
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 21
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 128
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 180
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 272
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 401
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 304
Avg New Search Result: 219.284060

```

图 3.4.19: $\epsilon = 0.1$ 在邻近集的平均值

可以发现，当 $\epsilon = 0.1$, $k = 20$ 时，拉普拉斯分布的方差非常大，那么这时候噪声会特别大，因此数据的可用性也很差。平均值也有着巨大不同，此时抗攻击效果很好。也就是说，即使取平均进行重复攻击，数据也被很好地保护了，如果我们隐私预算设置得**非常充足**（达到理论最好的值），那么安全性将在**任何时候**得到保障。

为防止随机性的影响，我对 $\epsilon = 0.1$ 又进行了实验，得到了如下结果：

```

Animals which carrots cost > 55 (Under DP): -32
Avg Old Search Result: 225.395728
=====Using neighbour dataset=====

```

图 3.4.20: $\epsilon = 0.1$ 平均值

```

Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 159
Avg New Search Result: 185.882669
=====

```

图 3.4.21: $\epsilon = 0.1$ 在邻近集的平均值

和上述的结果一样，保持了较大的方差，说明**隐私保护起到了良好的效果**。

4 实验心得

在完成这个关于交互式发布的差分隐私方案的实验中，我深刻体会到了隐私保护技术的重要性和实用价值。通过将拉普拉斯噪声添加到查询结果中，我们能够有效地保护数据集中个体的隐私，防止通过多次查询推断出敏感信息。这种方法在保证数据用途的同时，确保了个人数据的安全，平衡了数据可用性与隐私保护之间的矛盾。此外，实验中对隐私预算的分配和管理，以及对噪声量计算的精确控制，展示了差分隐私技术在现实应用中的复杂性和技术挑战，使我对数据科学和隐私保护领域有了更深的理解和认识。这次实验不仅增强了我的技术能力，也提升了我在处理真实世界数据问题时的敏感性和责任感。