Differential Privacy

Like k-Anonymity, *differential privacy* is a formal notion of privacy (i.e. it's possible to prove that a data release has the property). Unlike k-Anonymity, however, differential privacy is a property of *algorithms*, and not a property of *data*. That is, we can prove that an *algorithm* satisfies differential privacy; to show that a *dataset* satisfies differential privacy, we must show that the algorithm which produced it satisfies differential privacy.

A function which satisfies differential privacy is often called a *mechanism*. We say that a *mechanism* F satisfies differential privacy if for all *neighboring datasets* x and x', and all possible outputs S,

$$\frac{\Pr[F(x) = S]}{\Pr[F(x') = S]} \leq e^{\epsilon}$$

Two datasets are considered neighbors if they differ in the data of a single individual. Note that F is typically a *randomized* function, so that the probability distribution describing its outputs is not just a point distribution.

The important implication of this definition is that F's output will be pretty much the same, *with or without* the data of any specific individual. In other words, the randomness built into F should be "enough" so that an observed output from F will not reveal which of x or x' was the input. Imagine that my data is present in x but not in x'. If an adversary can't determine which of x or x' was the input to F, then the adversary can't tell whether or not my data was *present* in the input - let alone the contents of that data.

The ϵ parameter in the definition is called the *privacy parameter* or the *privacy budget*. ϵ provides a knob to tune the "amount of privacy" the definition provides. Small values of ϵ require F to provide *very* similar outputs when given similar inputs, and therefore provide higher levels of privacy; large values of ϵ allow less similarity in the outputs, and therefore provide less privacy.

How should we set ϵ to prevent bad outcomes in practice? Nobody knows. The general consensus is that ϵ should be around 1 or smaller, and values of ϵ above 10 probably don't do much to protect privacy - but this rule of thumb could turn out to be very conservative. We will have more to say on this subject later on.

The Laplace Mechanism

Differential privacy is typically used to answer specific queries. Let's consider a query on the census data, *without* differential privacy.

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
adult = pd.read_csv("adult_with_pii.csv")
```

"How many individuals in the dataset are 40 years old or older?"

In [6]:
```
adult[adult['Age'] >= 40].shape[0]
```

Out[6]:
```
14237
```

The easiest way to achieve differential privacy for this query is to add random noise to its answer. The key challenge is to add enough noise to satisfy the definition of differential privacy, but not so much that the answer becomes too noisy to be useful. To make this process easier, some basic *mechanisms* have been developed in the field of differential privacy, which describe exactly what kind of - and how much - noise to use. One of these is called the *Laplace mechanism*.

According to the Laplace mechanism, for a function $f(x)$ which returns a number, the following definition of $F(x)$ satisfies ϵ-differential privacy:

$$F(x) = f(x) + \text{Lap}(\frac{s}{\epsilon})$$

where s is the *sensitivity* of f, and $\text{Lap}(S)$ denotes sampling from the Laplace distribution with center 0 and scale S.

The *sensitivity* of a function f is the amount f's output changes when its input changes by 1. Sensitivity is a complex topic, and an integral part of designing differentially private algorithms; we will have much more to say about it later. For now, we will just point out that *counting queries* always have a sensitivity of 1: if a query counts the number of rows in the dataset with a particular property, and then we modify exactly one row of the dataset, then the query's output can change by at most 1.

Thus we can achieve differential privacy for our example query by using the Laplace mechanism with sensitivity 1 and an ϵ of our choosing. For now, let's pick $\epsilon = 0.1$. We can sample from the Laplace distribution using Numpy's `random.laplace`.

In [21]:
```
sensitivity = 1
epsilon = 0.1

adult[adult['Age'] >= 40].shape[0] + np.random.laplace(loc=0, scale=sensitiv
```

Out[21]:
```
14238.147613610243
```

You can see the effect of the noise by running the proceeding cell multiple times. Each time, the output changes, but most of the time, the answer is close enough to the true answer (14,235) to be useful.

How Much Noise is Enough?

How do we know that the Laplace mechanism adds enough noise to prevent the re-identification of individuals in the dataset? For one thing, we can try to break it! Let's write down a malicious counting query, which is specifically designed to determine whether Karrie Trusslove has an income greater than $50k.

In [28]:
```
karries_row = adult[adult['Name'] == 'Karrie Trusslove']
karries_row[karries_row['Target'] == '<=50K'].shape[0]
```

Out[28]: 1

This result definitely violates Karrie's privacy, since it reveals the value of the income column for Karrie's row. Since we know how to ensure differential privacy for counting queries with the Laplace mechanism, we can do so for this query:

In [29]:
```
sensitivity = 1
epsilon = 0.1

karries_row = adult[adult['Name'] == 'Karrie Trusslove']
karries_row[karries_row['Target'] == '<=50K'].shape[0] + \
  np.random.laplace(loc=0, scale=sensitivity/epsilon)
```

Out[29]: 2.198682025336349

Is the true answer 0 or 1? There's too much noise to be able to reliably tell. This is how differential privacy is *intended* to work - the approach does not *reject* queries which are determined to be malicious; instead, it adds enough noise that the results of a malicious query will be useless to the adversary.