Fan Zhang
3031922700
DS102 Project Part II

1 Bandits

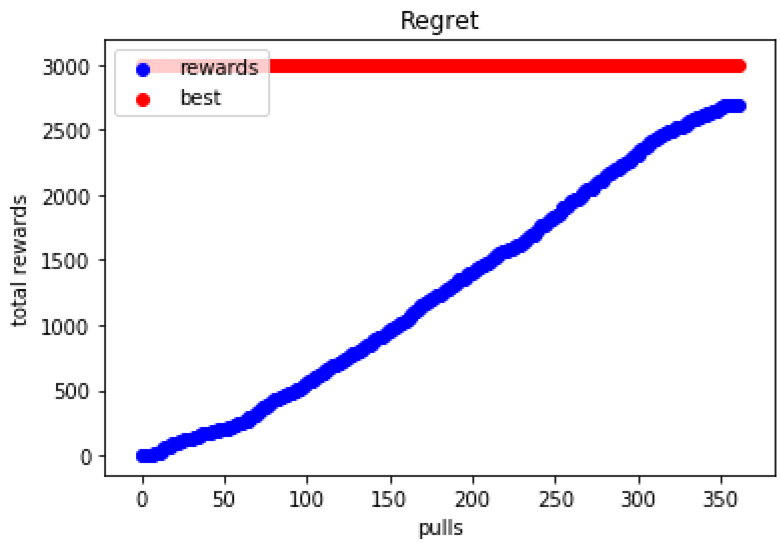
1.1 Formalizing the problem as a multi-armed bandits problem

Stations are the arms. Number of flyers given out at the street corner are the rewards. The rewards are modeled as bounded. The time horizon is the number of days for which the promotion will take place. The model assumes the rewards are independent and is a bounded distribution because its range from 0 to the population of the city. This distribution can be tested by plotting the data. Regret is defined as the most popular intersection of the whole time, which is the intersect that has maximum sum of counts.

1.2.1 Implementation and Results

At the start of the test, we don't know what is the best station(arm). We assume all of the stations have the same mean and upper bound. We begin with testing the first station on the first day. If there are people rent bikes in that station, we update the average of the first station its confidence bound. We compare this bound with the maximum bound of all and update the maximum bound. We repeat this process to every station at the given date. The algorithm then pulls, at each round t, the arm with the highest upper confidence bound. The rewards are subgaussian. For the confidence bound update, I used u_a +sqrt(4*var*log(n + 1)/N, where u_a is the current sample mean for arm a, N is the number of times arm a has been pulled up to and including iteration t. The goal is to find the arm A* with the highest mean reward μ* as fast as possible while maintaining performance.

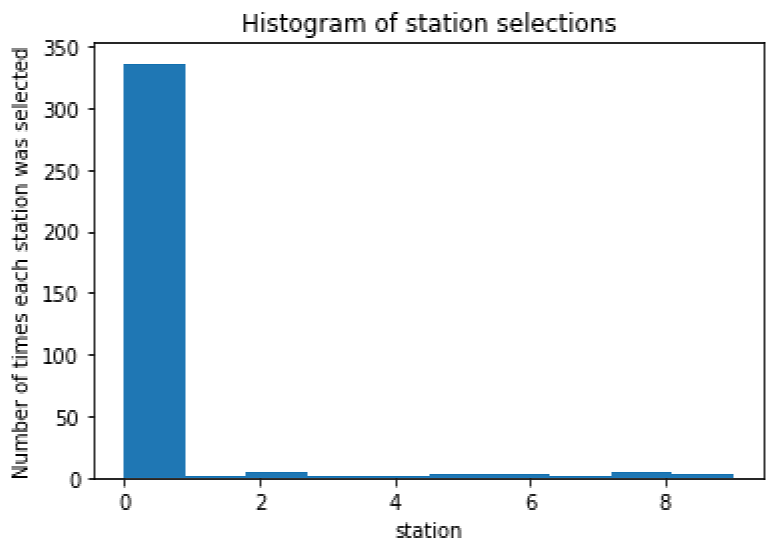
DC:
Estimate variance=3

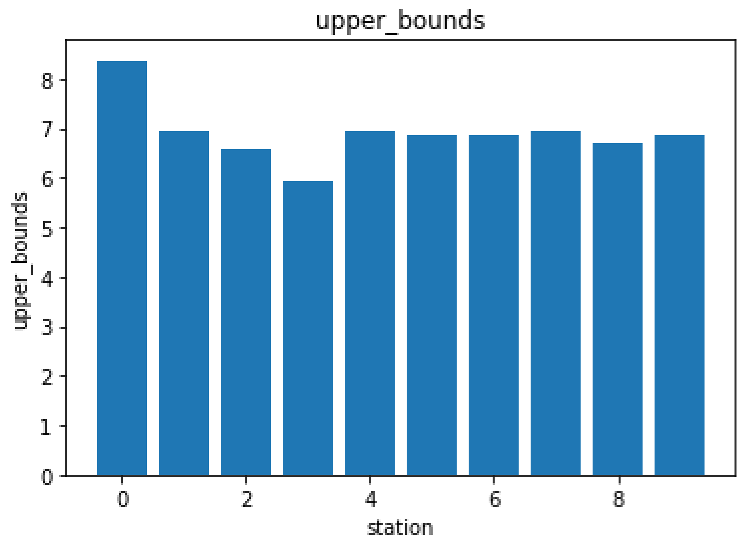


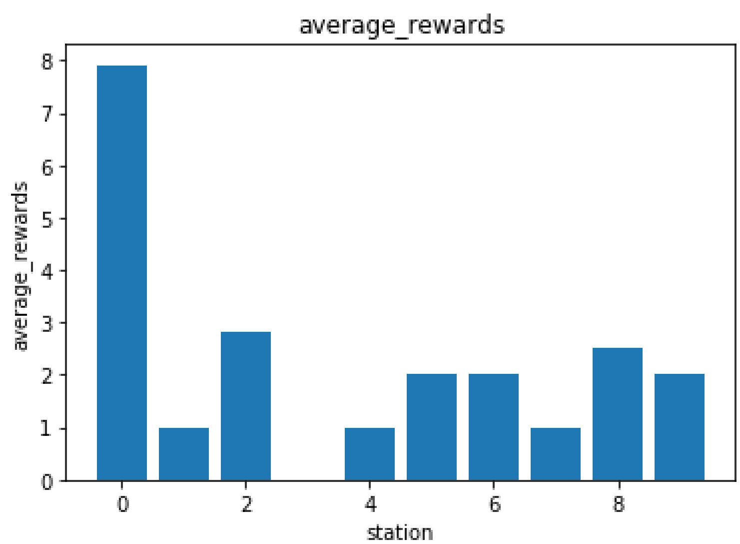
\

Station	True mean	Estimated Mean/average reward	Total pulls	Upper bound
Columbus Circle / Union Station	8.286567	7.91044776119403	336	8.36984298512098
Lincoln Memorial	7.857143	1	2	6.9455752682944505
Jefferson Dr & 14th St SW	7.532895	2.8	5	6.560311969555432
Massachusetts Ave & Dupont Circle NW	6.707692	0	2	5.9455752682944505
15th & P St NW	5.318182	1	2	6.9455752682944505
Thomas Circle	5.033784	2	3	6.854541878210866
14th & V St NW	4.902357	2	3	6.854541878210866

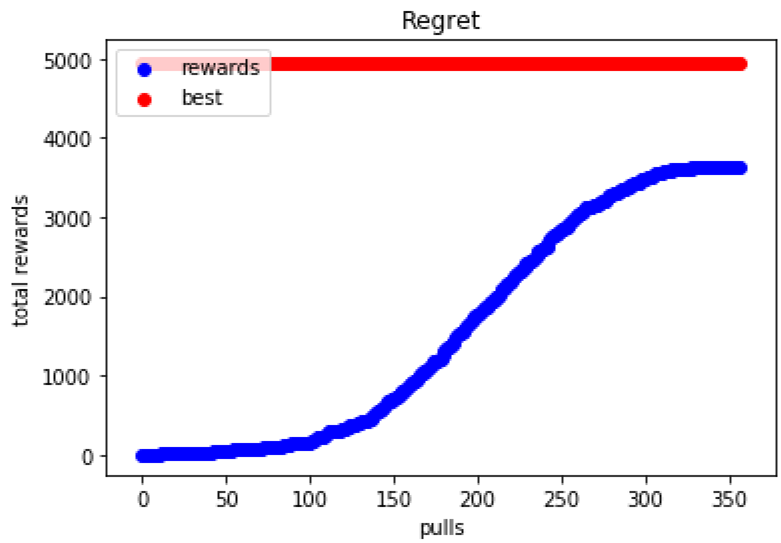
Jefferson Memorial	5.385214	1	2	6.9455752682944505
New Hampshire Ave & T St NW	4.607509	2.5	4	6.704156590266033
Eastern Market Metro / Pennsylvania Ave & 7th St SE	4.51773	2	3	6.854541878210866





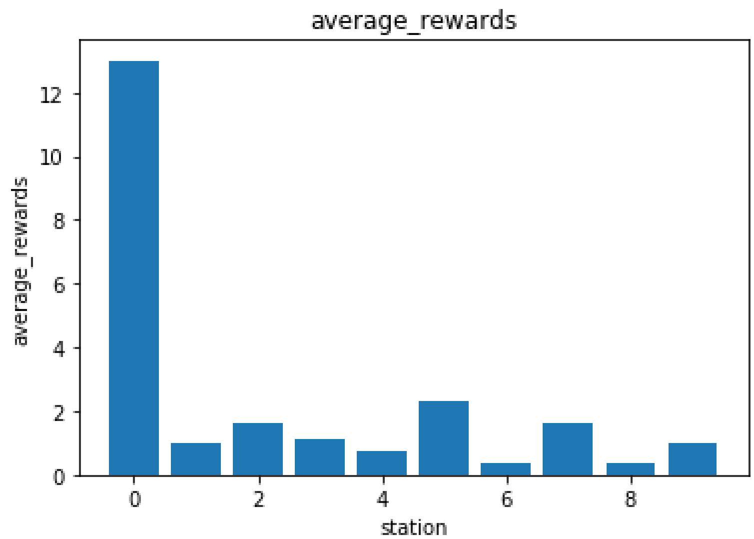


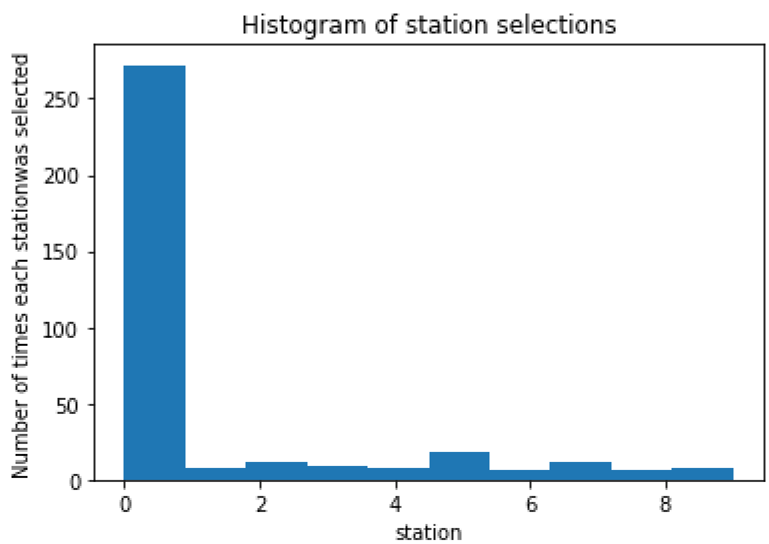
Chicago:
Estimate variance = 7

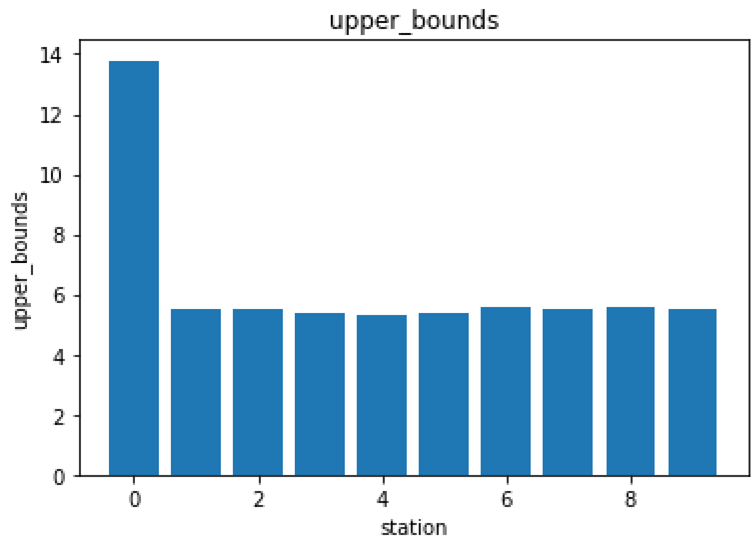


Station	True mean	Estimated Mean/average reward	Total pulls	Upper bound
Streeter Dr & Grand Ave	13.81203	13.00369003690037	272	13.782980569314828
Lake Shore Dr & Monroe St	8.390041	1	8	5.535644963643951
Lake Shore Dr & North Blvd	8.807512	1.6363636363636365	11	5.504374657898064
Theater on the Lake	8.017241	1.1111111111111112	9	5.387358192240777
Clinton St & Washington Blvd	7.140625	0.75	8	5.285644963643951
Clinton St & Madison St	5.536913	2.3333333333333335	18	5.357096642429301
Michigan Ave & Oak St	6.858369	0.3333333333333333	6	5.570645014750143
Millennium Park	6.195918	1.6363636363636365	11	5.504374657898064

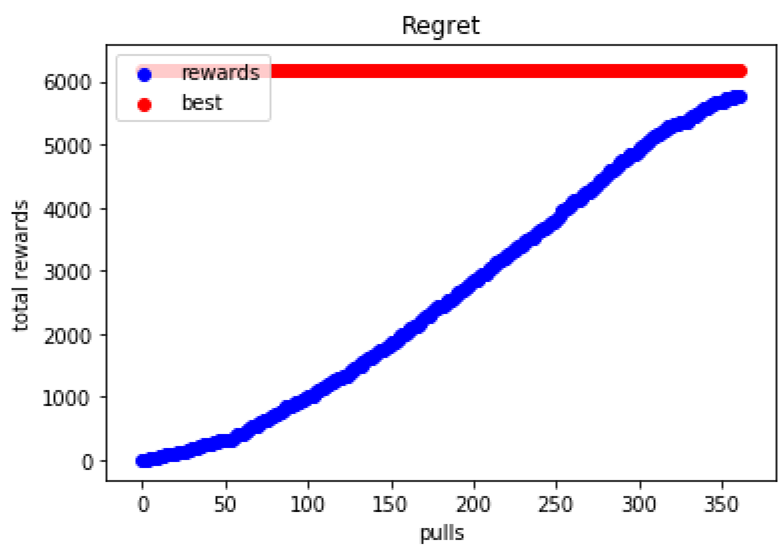
Canal St & Madison St	5.736434	0.3333333333333333	6	5.570645014750143
Canal St & Adams St	5.171315	1	8	5.535644963643951





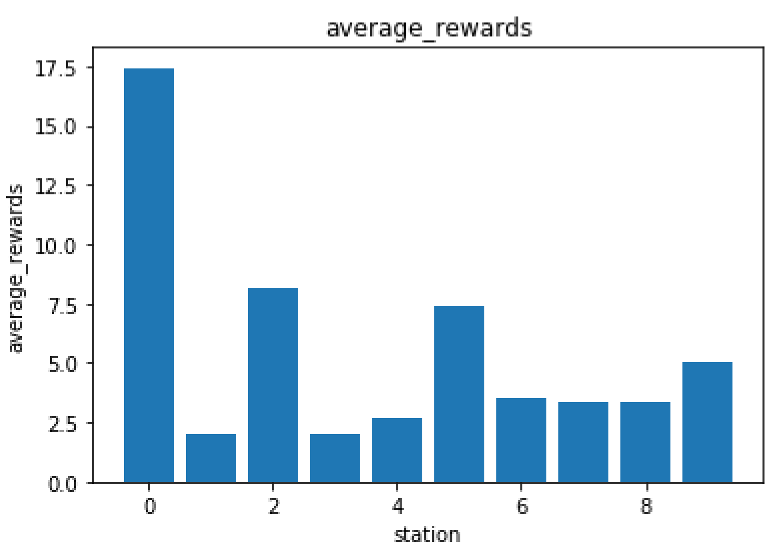


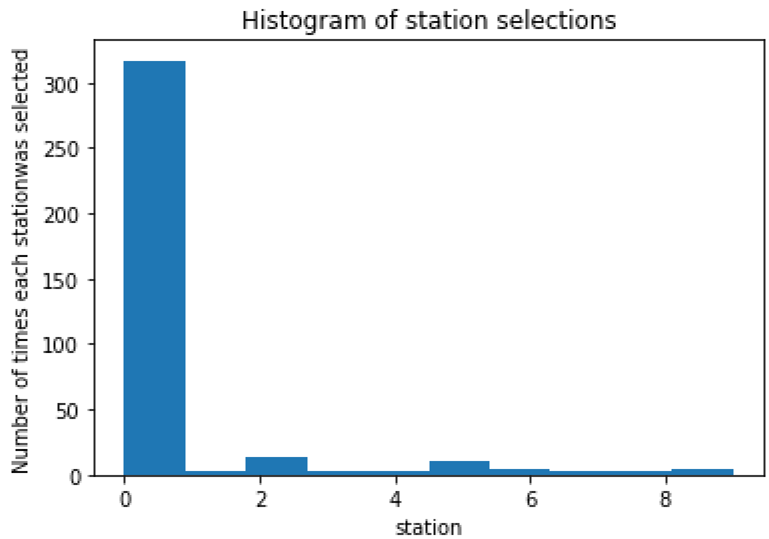
NY:
Estimate variance = 10

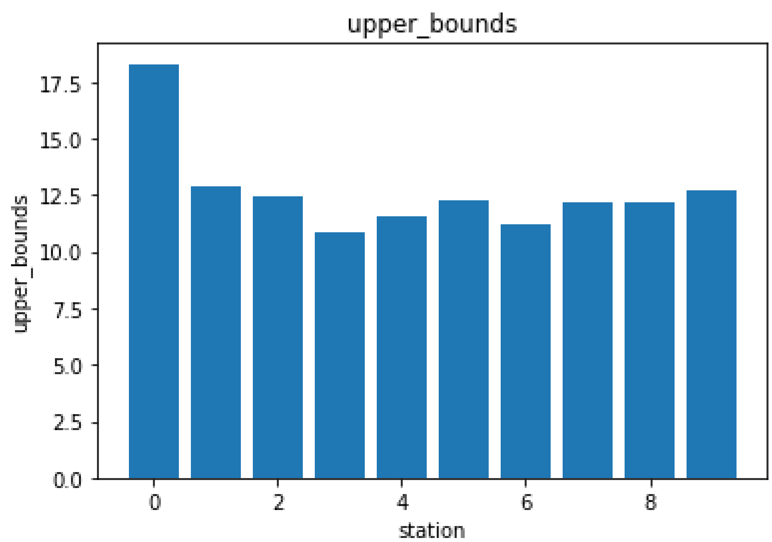


Station	True mean	Estimated Mean/average reward	Total pulls	Upper bound
Pershing Square North	17.057471	17.436708860759495	317	18.300293105829063
E 17 St & Broadway	11.713483	2	2	12.855085639299002
W 21 St & 6	11.280453	8.153846153846	13	12.41156103819

Ave		153		28
West St & Chambers St	11.476744	2	3	10.863140310165294
Broadway & E 22 St	11.087719	2.6666666666666665	3	11.52980697683196
8 Ave & W 33 St	9.822857	7.4	10	12.254541878210865
Broadway & E 14 St	9.601156	3.5	4	11.175704665909034
Cleveland Pl & Spring St	9.491329	3.3333333333333335	3	12.196473643498628
W 20 St & 11 Ave	9.712575	3.3333333333333335	3	12.196473643498628
Greenwich Ave & 8 Ave	9.402899	5	4	12.675704665909034







1.2.2 Discussion

If there is a change over different orderings of which arms to pull first and in what order, the algorithm can still pick the best arm but the number of pulls, average rewards and upper bound for each arm would be different. I have a time-varying δ=1/n^2 in my model to ensure that each arm will always be pulled. The problem is, my result is very sensitive to the initialization. If I changed my estimate variance, the result would be totally different. The variances are different significantly for each arm.

If instead of sending people to the same station in all year, we could further explore seasonality of our data. There could be some time where a particular station has a traffic peak. If we can find those special time period and send people to a different station, the regret would be better than the current regret.

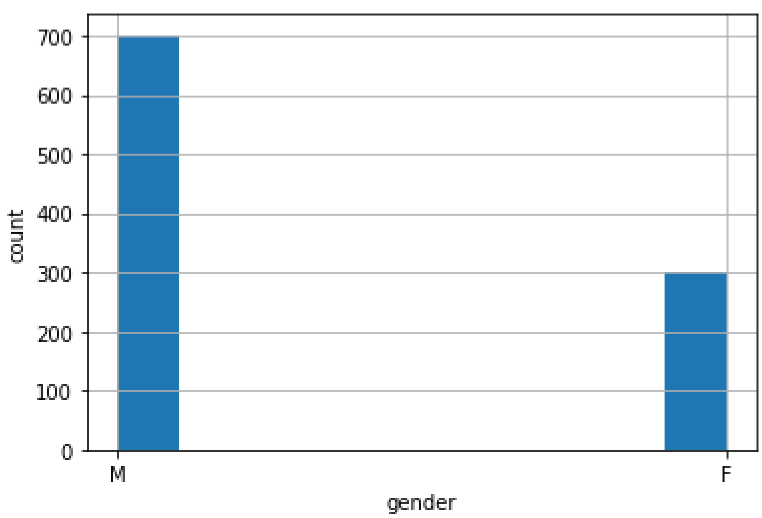
1.3 Takeaways

Using UCB to solve the problem depends on several conditions. First, we need to make sure that we have a reasonable estimation of the model. If the model is correct, UCB can save us a lot of time to get the optimal station. However, for example, if the variance we estimate is wrong, we could get a suboptimal result. Our data size is too small, therefore initialization of model variance matters a lot to results. Exploration could take a lot of time and it is not very time and cost efficient for a startup. For applicability, this formulation violates independence of the rewards for each pull of the arms because if you went flyering at one station, you would expect the same group of people to be at the same station tomorrow. They have taken your flyer, so on the next day, they probably won't take it again. The rewards are not independent in this case. Also, our regret is not very accurate because if you keep flyering the whole year at the same time, the group of people you covered is limited. Ultimately, I don't recommend using UCB to adaptively place the person handing out promotional flyers. Our case violate UCB assumptions. I would suggest the company should have a schedule for flyering at different stations because it is more cost and time efficient.

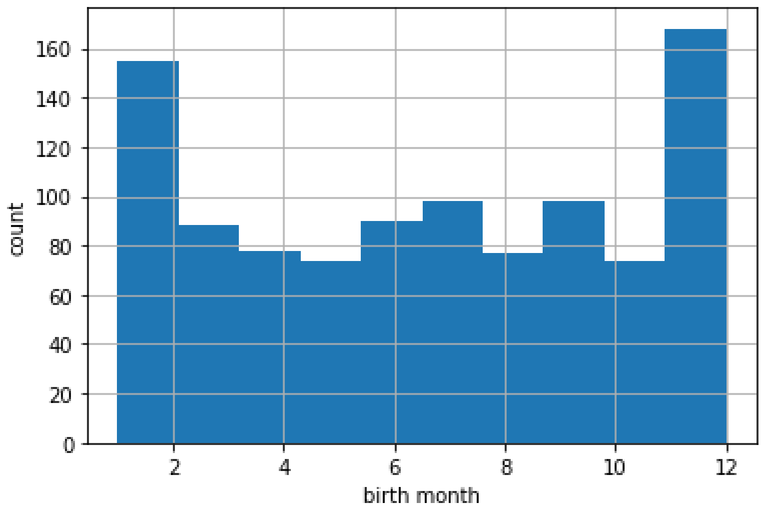
2 Privacy Concerns

2.1 Exploratory Analysis

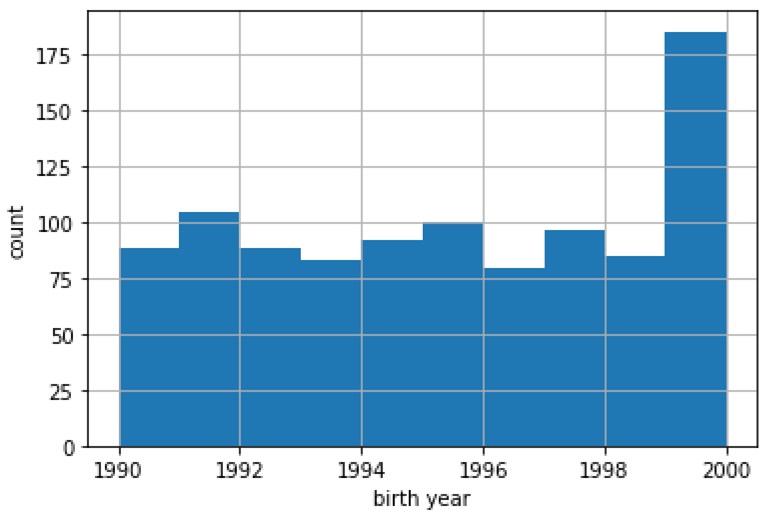
- Number of females and the number of males in leaked.csv.



- Distribution of birth months in leaked.csv.



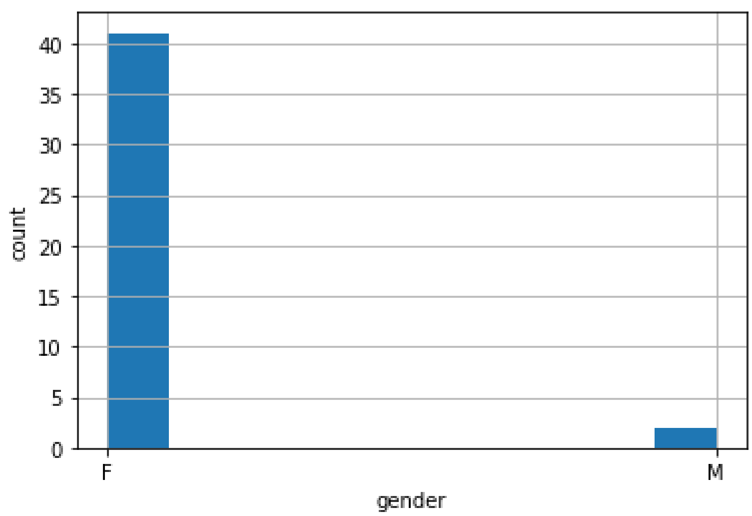
• Distribution of birth years in leaked.csv.



Birth month is uniformly distributed from February to November. Birth year distribution is uniformly distributed from 1990 to 1999. The gender distribution is similar to the rental bike datasets.

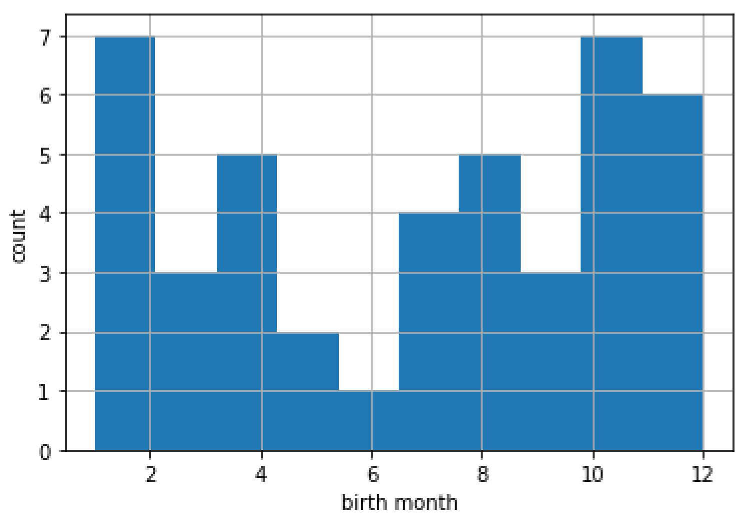
2.2 Simple Proof of Concept

• 43 users can be isolated based on just those three attributes.
• Number of females and the number of males in the set of identifiable users



• The distribution of gender in the set of identifiable users does not match the corresponding distribution from the leak dataset. Far more female compared to male in identifiable users. Whereas in the leak dataset, there are roughly two times more male than females. This could be due to that in the leak dataset, more male have repeated information because the group size is bigger. For female, because there are fewer samples, most of them are unique.

• Distribution of birth months for identifiable users.



• The distribution of birth month in the set of identifiable users does not match the corresponding distribution from the leak dataset. The size of identifiable users are too small to show a meaningful distribution.

• Extract the scooter rentals each identifiable user have made from berkeley.csv:
berkeley['idu'] = berkeley['sex']+berkeley["month"].map(str) +berkeley['year'].map(str)
identifiable['idu'] = identifiable['sex']+identifiable["month"].map(str) identifiable['year'].map(str)

pd.merge(identifiable,berkeley,on="idu")

2.3 A More Elaborate Attack

• Parameters p1 and p2 using the trips made by the set of identifiable users.
p1 = 0.12093023255813953
p2 = 0.2837209302325581
After merge leaked data and berkeley data on birth month, year and gender, I checked the new table by finding the entries that satisfied [zip != start] divided by the total number of entries in the merged table, to find p1. And the same process with condition [zip != end] to find p2.

• 95% confidence intervals around both p1 and p2:
By normal approximation of the binomial distribution:
95% interval for p1:
p1+1.96*(sqrt(p1*q1/n))=0.16451311577806205
p1-1.96*(sqrt(p1*q1/n))=0.077347349338217
95% interval for p2:
p2+1.96*(sqrt(p2*q2/n))=0.3439801715412996
p2-1.96*(sqrt(p2*q2/n))=0.22346168892381663
.

• There are 825 theoretically identifiable users. They are either uniquely identifiable in Berkeley dataset by starting zip code, birth month, year and gender or uniquely identifiable in Berkeley dataset by ending zip code, birth month, year and gender.

• Algorithm:

```
def find(i):
  bkl['idu'] = bkl['sex']+bkl["month"].map(str) +bkl['year'].map(str)+bkl['start'].map(str)
  result=df1[df1['idu']==bkl.iloc[i]['idu']]
  if len(result) != 0:
    return result.sample(n=1, random_state=1)
    bkl['idu'] = bkl['sex']+bkl["month"].map(str) +bkl['year'].map(str)+bkl['end'].map(str)
    result1=df1[df1['idu']==bkl.iloc[i]['idu']]
  if len(result1) != 0:
    return result1.sample(n=1, random_state=1)
  else:
    bkl['idu'] = bkl['sex']+bkl["month"].map(str) +bkl['year'].map(str)
```

```
df1['idu'] =df1['sex']+df1["month"].map(str) +df1['year'].map(str)
result2=df1[df1['idu']==bkl.iloc[i]['idu']]
return result2.sample(n=1, random_state=1)
```

First assume start location's zip code is the actual zip code of the user because (1-p1) is greater than (1-p2). Use zip, gender, birth month and year to match with the leaked database. If there are multiple matches, randomly pick one. If there is no matches, use end location's zip code as the real zip code to repeat this procedure. If there is still no matches, then exclude the zip code from the berkeley dataset to only find matches from gender, birth month and year.

2.4 Takeaways

Privacy is easy to invade. From our example, many users in Berkeley dataset can be uniquely identified by merging with the leaked dataset that contain overlapping information. To improve the privacy of already released data, we could add noise to the true answer .The goal is to minimize the magnitude of this noise, while maintaining differential privacy (for a fixed privacy level). We can also randomize our data before release or only respond to query but not release the whole dataset.