Causes and Solutions to U.S. Opioid Crisis

Shuo Liu[1], Rui Xi[1], Rui Yang[1], Yuqi Wu[1]

SDCS, SYSU

Summary

With the opioid crisis becoming a national concern, the federal government is struggling to figure out how to respond. Why does it happen? How does it evolve? And what can we do to deal with it? Here, we proposed model `OCEAN` and its enhancement `D-OCEAN` to delve into the evolution of opioids.

We first get the growth pattern of drug reported quantity through a large number of data analysis and case studies. We observe that: **(a)** A county owns a similar growth pattern with its nearby. **(b)** Linear growth patterns can be found in counties who are isolated by similarity with the surrounding. Based on such observations, we assume that the increase in drug reports is made up of two main components: **intrinsic increment** and **extrinsic influence** from nearby counties. Then we defined a cellular automata to simulate evolution of opioid cases for all counties. Reverse evolving, origin of a specific opioid can be traced. Forward evolving, prediction is made for each location.

After that, we further consider the **socio-economic factors** and modify our model. By computing the Pearson correlation coefficient between drug reported quantity and the socio-economic factors, we find some important **demographic features** which are highly related to opioid use. **K-means** algorithm is then applied to group data into several groups, each of which represents a typical type. Our results have been reversely verified to ensure the accuracy of similarity analysis. Taking the idea of simulate anneal, we add a new update rule to make intrinsic factor more considerate. As proved by comparative experiments, enhanced model possesses a higher robustness and more accurate predict results.

Base on our works above, we present some strategies for countering the opioid crisis. We validate that such strategies can really improve the opioid epidemic situation and sensitivity analysis shows the utility of each strategy.

In conclusion, we employ a cellular automata for simulating the evolution patterns of opioids in **Task I**. We enhanced our model in **Task II** by introducing some socio-economic factors, taking account of demographic heterogeneity in each state. Finally, corresponding solutions were put forward, according to the analysis of our models.

Contents

1 Problems

1.1 Background

The United States is experiencing a national crisis regarding the use of synthetic and non- synthetic opioids, either for the treatment and management of pain (legal, prescription use) or for recreational purposes (illegal, non-prescription use). Federal organizations such as the Centers for Disease Control (CDC) are struggling to save lives and prevent negative health effects of this epidemic, such as opioid use disorder, hepatitis, and HIV infections, and neonatal abstinence syndrome.1 Simply enforcing existing laws is a complex challenge for the Federal Bureau of Investigation (FBI), and the U.S. Drug Enforcement Administration (DEA), among others.

There are implications for important sectors of the U.S. economy as well. For example, if the opioid crisis spreads to all cross-sections of the U.S. population (including the college-educated and those with advanced degrees), businesses requiring precision labor skills, high technology component assembly, and sensitive trust or security relationships with clients and customers might have difficulty filling these positions. Further, if the percentage of people with opioid addiction increases within the elderly, health care costs and assisted living facility staffing will also be affected.

The DEA/National Forensic Laboratory Information System (NFLIS), as part of the Drug Enforcement Administration's (DEA) Office of Diversion Control, publishes a data-heavy annual report addressing "drug identification results and associated information from drug cases analyzed by federal, state, and local forensic laboratories." The database within NFLIS includes data from crime laboratories that handle over 88% of the nation's estimated 1.2 million annual state and local drug cases. For this problem, we focus on the individual counties located in five (5) U.S. states: Ohio, Kentucky, West Virginia, Virginia, and Tennessee. In the U.S., a county is the next lower level of government below each state that has taxation authority.

1.2 Dataset

We collect and download several datasets from DEA and other American drug administration. The first file (MCM_NFLIS_Data.xlsx) contains drug identification counts in years 2010-2017 for narcotic analgesics (synthetic opioids) and heroin in each of the counties from these five states as reported to the DEA by crime laboratories throughout each state. A drug identification occurs when evidence is submitted to crime laboratories by law enforcement agencies as part of a criminal investigation and the laboratorys forensic scientists test the evidence. Typically, when law enforcement organizations submit these samples, they provide location data (county) with their incident reports. When evidence is submitted to a crime laboratory and this location data is not provided, the crime laboratory uses the location of the city/county/state investigating law enforcement organization that submitted the case. For the purposes of this problem, we can assume that the county location data are correct as provided.

The additional seven (7) files are zipped folders containing extracts from the U.S. Census Bureau that represent a common set of socio-economic factors collected for the counties of these five states during each of the years 2010-2016 (ACS_xx_5YR_DP02.zip). *(Note: The same data were not available for 2017.)* All of the datasets we used have been push on the transfer gate, you can download them via this link.

2 Introduction

With the nationwide opioid epidemic, making good use of them and preventing their negative health effects have become one of the urgent missions for the U.S. Centers of Disease Control (CDC). [1]

In addition to some synthetic opioids used for medical anesthesia, large quantities of addictive heroin and other non-synthetic opioids have entered the market, seriously jeopardizing the health of American citizens and bringing huge economic and social pressure to the U.S. government.[2] Despite legislative, law enforcement, and judicial efforts, the trend in opioid abuse does not appear to have abated so far.[3,4] As the number of illnesses and deaths increases, the opioid crisis is shaping up to be a catastrophe for American society.[5]

As the opioid crisis has been greatly concerned, the National Forensic Laboratory Information System (DEA/NFLIS) collects fairly comprehensive data from crime laboratories around the country, hoping to find effective coping strategies by analyzing it. Here, we are expected to help NFLIS identify the causes of the opioid crisis and to propose practical solutions. Specifically, we mainly focus on individual counties located in five U.S. states (VA, OH, PA, WV, KY), and set the following goals:

- Describe spread and characteristics of opioid incidents.
- Speculate on the origins of opioids and predict their future.
- Give analysis of the correlation between opioid use and socio-economic factors.
- Find possible strategies to deal with the opioid crisis and analyze their feasibility.

Many existing models have shed lights on the pattern of opiate addiction transmission. Early in 1997, D.R. Mackintosh, G.T. Stewart proposed an exponential model to illustrate how the use of heroin spreads in epidemic fashion.[6] Based on the principles of epidemiology, Emma White and Catherine Comiskey present an ODE model of opiate addiction in 2007.[7] G.P. Samanta modified the White and Comiskey heroin epidemic model into a non-autonomous heroin epidemic model with distributed time delay.[8] However, to our best knowledge, most of the existing models are small-range transmission models based on epidemiological principles, no model uses finite automata to simulate large-scale drug transmission. In addition, these models only mention non-prescriptive opioids like heroin, but prescription drugs may have very different patterns in transmission to some extent. More importantly, given the heterogeneity of states, many studies inspire us to take demographic structure into account.[9,10]

In this work, we proposed a novel model called `OCEAN` and its enhancement `D-OCEAN` to cope with the opioid crisis. The framework of `OCEAN` and `D-OCEAN` is shown in **Figure** 1.

Our framework in **Figure** 1 has the following attributes:

- **2 Models:** `OCEAN` and `D-OCEAN` are involved in our framework to solving problem in **Task I** and **Task II** respectively.
- **3 Layers:** Socio-economic factors, drug identification as well as download map are input from input layer to model layer for modeling, and then generate the final results.
- **4 Update Rules:** Our models follow 4 update rules to estimate the drug reported quantity and increment in the next year.
- **5 Techniques:** We mainly apply 5 techniques to accomplish our framework: cosine similarity analysis, Pearson similarity analysis, cellular automata simulation, K-means algorithm and simulate anneal.

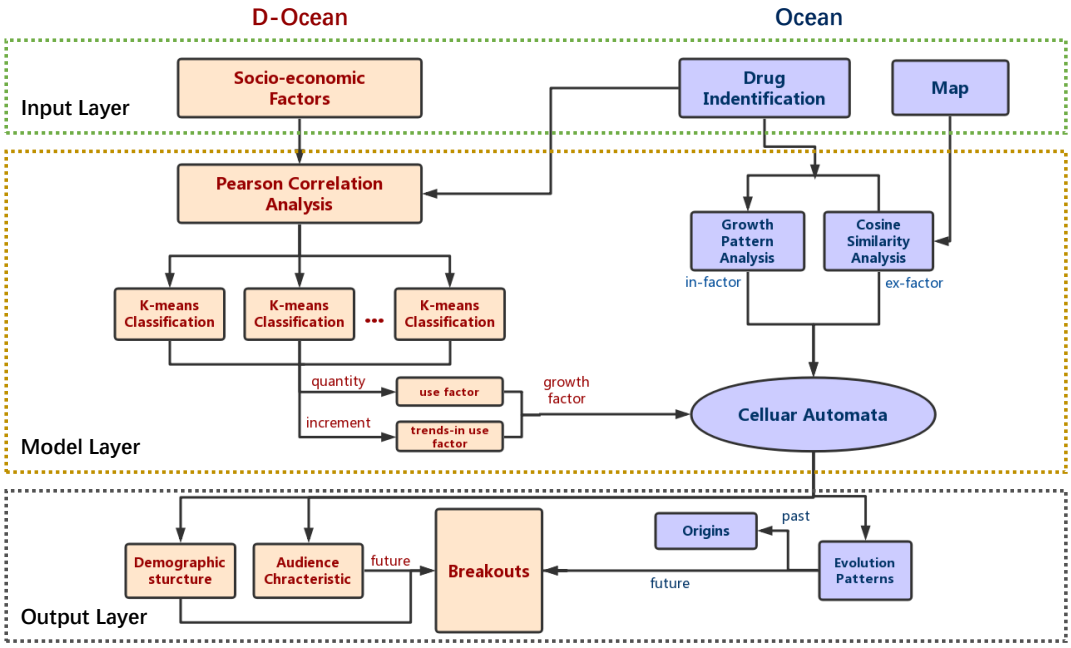

Figure 1: Framework of our model: `OCEAN` and `D-OCEAN`

3 Assumptions and Definitions

3.1 General Assumptions

Assumption 1 *The effect of irresistible factors like warfare and nature disaster will not be taken into consideration in this problem.*

Upheaval of situations cause by inevitable cases are supposed to be neglected for difficulty of prediction.

Assumption 2 *The provided data is accurate and reliable so that we can obtain correct information from it.*

The accuracy of the data is the cornerstone of the experiment.

Assumption 3 *There is drug flow from county to county.*

In real life, proximity facilitates the spread of drugs.

Assumption 4 *There is a correlation between drug cases and demographics features.*

The heterogeneity of county population is an important cause of event deviation.

3.2 General Definitions

- **Drug reports/opioid cases:** reported opioid incidents in counties. We mainly focus on two aspects of them: the quantity and the increment.

- **Evolution:** sum of all forms of motion in a given space-time. In this situation, evolution represents spread and characteristics of opioid incidents.

- **Opioid-increment island:** county whose incremental trend is not similar to all its surrounding counties. The evolution of opioid-increment islands is relatively isolated.

- **Cellular automaton:** local grid dynamics model with discrete time, space and state, which are able to simulate spatio-temporal evolution of complex systems. [11]

- **Demographic features:** socio-economic features (e.g. fertility, educational attainment, marital status) of the population. Note that demographic features can be directly extract from data given.

- **Significant characteristics:** some prominent characteristics that mined from demographic statistics, e.g. poor family relations, overseas, educational level etc. Significant characteristics are not intuitive information and can only be obtained by data mining.

- **K-means algorithm:** method which builds k classes with minimal distance among the sample observations. [12]

- **Cluster score:** correlation between the drug reported quantity and a particular significant characteristics. A cluster with high score is more likely to have a strong relation with opioid cases.

- **Use factor:** factor measuring the number of crowds who have the highly correlated demographic features. When a county has a large use factor, it may be closely linked to the drug incidents.

- **Trends-in-use factor:** factor reflecting the level of outbreak vitality. When a county has a large trends-in-use factor, it may have a tremendous explosive potential with respect to certain opioids.

4 Task I: Model Evolution of Opioid Cases

4.1 Model Description

In this part, we build a grid dynamics model named **National Opioid-cases Evolution Cellular Automaton** (`OCEAN`), which serves to describe the spread and characteristics of opioid cases in five U.S. states. The objective of this part of model is to trace the origin of opioids and to predict potential future outbreak sites in each state.

Firstly, obtaining the geographical coordinates and distances among counties, we use **cosine similarity** formula to get the correlation of opioid cases between any two counties. A lot of hidden information is mined and put forward to make some basic assumptions. Based on these

assumptions, we take advantage of a discrete, stochastic **cellular automata**, effectively simulating the evolution of reported synthetic opioid and heroin incidents.

By performing reverse evolution on cellular automata, we obtained the origin of specific opioids in each state. Moreover, results of the evolution allow us to predict when and where a particular opioid might flood. `OCEAN` will help to identify potential drug abuse in future for prevention.

4.2 Model Assumption

Assumption 5 *There are interactions between closer counties about opioid incidents within a certain range.*

By comparing the similarity of the opioid reports increment in different counties of five states, we find that some of the closest cities had similar growth patterns between 2010 and 2018. This gives us reason to believe that there may be some interactions between nearby cities.

Assumption 6 *Growth patterns of the opioid-increment islands are nearly linear.*

Opioid-increment island refers to a county where the incremental trend is not highly similar to all its surrounding cities. Through a large number of data analysis and case studies, we find that growth patterns of these opioid-increment islands are approximately linear.

Assumption 7 *Estimate of the drug reports for a county in the next year is calculated by the combination of intrinsic and extrinsic factor.*

According to the assumptions above, we put other factors on the back burner and only focus on two growth-factors, *i.e.* intrinsic factor and extrinsic factor. In other words, update rules of `OCEAN` mainly take these two factors into consideration.

4.3 Mathematical Notations

In this part, we use the notations in **Table** 1 to present the indicators in our `OCEAN`.

4.4 Model Details

Problems

The description of spread and characterization of opioids can be attributed to the evolution of a complex systems. Discrete, stochastic cellular automata, having the ability to simulate the spatio-temporal evolution of complex systems, naturally becomes our first choice.

In this problem, we are given drug identification counts $R^{(c_n,t)}$ in year 2010-2017 for narcotic analgesics and heroin in each of the counties. Our objective is to use these data to simulate the evolution of reported synthetic opioids and heroin between different counties over time.

Methods

We first visualized the data as **Figure** 2 to help to have a better overview on the synthetic opioid and heroin distribution.

Because of the our assumption that some extrinsic factors may participate in the opioid evolution, so it is necessary to consider the correlation of geographical effects. After getting the

Notations	Descriptions
t	Year of drug reports
c_n	County n in VA, OH, PA, WV, KY
$\mathcal{NB}$	Nearby county collection
R, ΔR	drug reported quantity and annual increment
ΔR_{in}, ΔR_{ex}	Intrinsic and extrinsic growth quantity
$\hat{R}$, $\Delta\hat{R}$	Estimate of drug reported quantity and increment
k_{in}, k_{ex}	Weight of intrinsic and extrinsic opioid growth quantity
$S(c, c')$	Similarity of opioids' growth pattern between c and c'
α, β	Incremental parameter of intrinsic and extrinsic opioid growth

Table 1: Mathematical Notations

increment ΔR of drug reports in each county with the previous data, we decide to use cosine similarity to analyze the influence between counties. Specifically, the similarity of growth modes of county c and county c' can be expressed by the following formula.

$$S(c, c') = \frac{\Delta\mathbf{R}^{(c)}\Delta\mathbf{R}^{(c')}}{\|\Delta\mathbf{R}^{(c)}\| \cdot \|\Delta\mathbf{R}^{(c')}\|} \tag{1}$$

Added inherent growth into consideration, we employ cellular automaton $A = (L, S, N, f)$ to simulate the opioid evolution of counties, where L is cellular space - counties in five states; S is state of the cellular - drug reports; N is neighbor connections - nearby county; f is the update rules - definition of evolutionary mode. The `OCEAN` metric comes as follows.

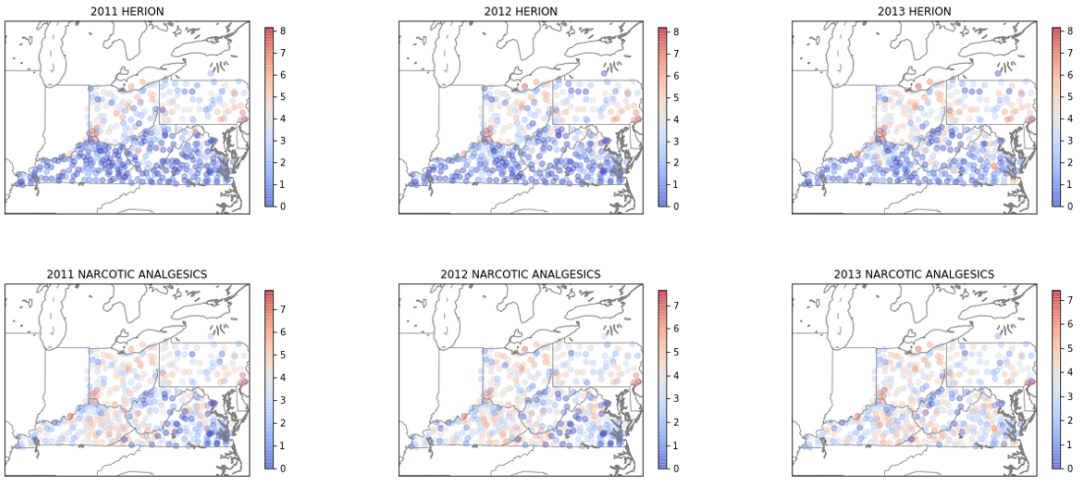

Figure 2: Overview on the distribution of synthetic opioid and heroin.

Metric 1: OCEAN Model

We estimate opioid cases quantity $\hat{R}^{(c_n,t)}$ of c_n in t by its estimate increment $\Delta\hat{R}^{(c_n,t)}$.

$$\hat{R}^{(c_n,t)} = R^{(c_n,t-1)} + \Delta\hat{R}^{(c_n,t)} \tag{2}$$

The estimate increment $\Delta\hat{R}^{(c_n,t)}$ is the evaluation of internal and external factors.

$$\Delta\hat{R}^{(c_n,t)} = k_{in}^{(c_n)}\Delta\hat{R}_{in}^{(c_n,t)} + k_{ex}^{(c_n)}\Delta\hat{R}_{ex}^{(c_n,t)}, \quad k_{in}^{(c_n)}\&k_{ex}^{(c_n)} \sim \mathcal{N}(0.5, 0.1) \tag{3}$$

The estimate of intrinsic increment is calculated by multiplying incremental parameter α with the increment of last year.

$$\Delta\hat{R}_{in}^{(c_n,t)} = \alpha \cdot \Delta R^{(c_n,t-1)}, \quad \alpha \sim \mathcal{N}(1.0, 0.2) \tag{4}$$

The influence of external factors is mainly reflected in the contribution of weighted similarity between c_n and its nearby counties $\mathcal{NB}(c_n)$.

$$\Delta\hat{R}_{ex}^{(c_n,t)} = \sum_{c_i \in \mathcal{NB}(c_n)} \beta_i \cdot S(c_n, c_i)\Delta R^{(c_i,t-1)}, \quad \beta_i \sim \mathcal{N}(0.3, 0.1) \tag{5}$$

5 Task II: Model Demographic Factors Impact on Opioid Use

5.1 Model Description

In this part, we build an enhanced model named **Demographics-based National Opioid-cases Evolution Cellular Automaton** (`D-OCEAN`), which analyzes the relationship between opioid use and socio-economic factors and describes the evolution pattern of opioid cases in more accurate manner. The objective of this part of model is to draw a parallel between specific drug use and certain demographic characteristics and increase supervisory strength to some crowds accordingly.

At the beginning of this model, we use **Pearson correlation** coefficient to screen out some features with high correlation. **K-means** algorithm is applied here to classify the demographic features of each city. We calculate the score of all clusters according to the quantity of their average drug reported quantity and normalize the weighted sum for each county as use factor. Replacing quantity with increment, we get the use-of-trend factor in a similar way. Finally, we stand on the shoulders of our previous work and modify `OCEAN` to a more complete model `D-OCEAN` taking the idea of **simulate anneal** algorithm.

When observing **K-means** algorithm results, it's understandable for us to discover some implied connections of opioid use and social structure. The results tell us about the citizen construction of a county, the characteristics of people who are prone to opiate addiction, as well as high-risk areas. All of these insights provide guidance for the government's macro-regulation.

5.2 Model Assumption

Assumption 8 *There will be no dramatic changes on the socio-economic characteristics of a county in the next few years.*

Dramatic change of socio-economic characteristics will add errors and uncertainty when we predict the trend. Because we need to predict the characteristics of each county in the future as well.

Assumption 9 *We only consider the socio-economic factors that can be quantified using the provided data.*

We are only allowed to use the given data. Thus, we will not take other socio-economic factors into consideration.

Assumption 10 *The drug reported quantity of a county won't keep growing for many years.*

In reality, if the drug reported quantity of a county has kept growing for several years, the government will take some measures to impede the spread.

5.3 Methemtical Notations

In this part, we use the notations in Table tab1e 4.3 to present the indicators in our D-OCEAN.

Notations	Descriptions
$X_i^{(c_n)}$	Demographic feature i of c_n
$\overline{R_{X_i,k}}$	Mean drug reported quantity of cluster k for X_i
$\widehat{R}$, $\Delta\widehat{R}$	Enhanced estimate of quantity and annual increment
ρ	Pearson correlation coefficient of two variables
E, μ, σ	Sample expectations, mean and standard deviation
$C_{X_i,k}$	Cluster k for the feature X_i
$M_{X_i,k}$	Center of cluster k for the feature X_i
$Score_{X_i,k}$	Score of cluster k in X_i
$a^{(c_n)}$, $b^{(c_n)}$	Use and trends-in-use factor for c_n
τ	Times of growth in drug reported quantity
λ	Growth factor of estimate incremental $\Delta\widehat{R}$

Table 2: Mathematical Notations

5.4 Model Details

Problems

This part of model serves to analyze the impact of demographic factors (e.g. fertility, educational attainment, marital status) on opioid use and the trend of use.

The extraction of significant demographic characteristics could be viewed as a K-means problem, which helps us to define the evolution model more reliably. As one of the probability-based algorithm, simulate anneal avoids results falling into local optimal solutions and make our new model more reliable.

Given the common sets of socio-economic factors collected for the counties, our goal can be split into two parts: for every important demographic feature, to minimize the total distance from the significant characteristic center to sample observation; to modify the update rules, using relation with demographic factors.

Methods

Based on historical data of socio-economic factors, we first identify the highly correlated demographic features calculating by Pearson correlation coefficient.

$$\rho^{R,X_i} = \frac{E[(R-\mu_R)(X_i-\mu_{X_i})]}{\sigma_R - \sigma_{X_i}} \tag{6}$$

For each of these highly correlated demographic features, we adapt the K-means algorithm. Specifically, given points of demographic feature i of c_n, *i.e.* $\{X_i^{(c_1)}, X_i^{(c_2)} \cdots\}_{\forall c_n}$, we are supposed to classify the data in to k clusters. The algorithm goes as follows:

1. Randomly initialize the cluster center $M_{X_i,1}, M_{X_i,2}, \cdots, M_{X_i,k}$.

2. For each point $X_i^{(c_n)}$, assign it to the cluster $C_{X_i,j}^{(c_n)}$ with a minimal distance.

$$C_{X_i,j}^{(c_n)} = \arg\min_j \|X_i^{(c_n)} - M_{X_i,j}\|^2 \tag{7}$$

3. For the cluster $C_{X_i,j}^{(c_n)}$, update its center $M_{X_i,j}$ to minimize the total distance from the center to every point in it.

$$C_{X_i,j}^{(c_n)} = \arg\min_j \sum_{\forall c_n} \|X_i^{(c_n)} - M_{X_i,j}\|^2 \tag{8}$$

4. Repeat step 2 and 3 until convergence.

Next, the mean drug reported quantity in all clusters $\{\overline{R_{X_i,1}},\overline{R_{X_i,2}},\cdots,\overline{R_{X_i,k}}\}_{\forall k}$ are calculated. Sorted from large to small, the scores of them $\{Score_{X_i,1},Score_{X_i,2},\cdots,Score_{X_i,k}\}_{\forall k}$ corresponding to X_i are generated in order.

The use factor of each county $a^{(c_n)} \in [0,1]$ is then obtained after normalizing the Pearson-weighted sum of these scores. Substituting quantity with increment, we use the same technique to get the trends-in-use factor of c_n, *i.e.* $b^{(c_n)} \in [0,1]$.

Finally, the update rules of cellular automaton we employed in previous are modified taking the idea of simulate anneal. We generate a random float r and compare it to the threshold $(b^{c_n})^{\tau/a^{(c_n)}}$, which is calculated from the above two factors. Counties with a large number of strongly related features crowds (use factor) and high level of outbreak vitality (trends-in-use factor) are more likely to grow in the way above. By applying this thought in our prior works - OCEAN, our new model become more considerable and robust.

Metric 2: D-OCEAN Model

We estimate opioid cases quantity $\widehat{R}^{(c_n,t)}$ of c_n in t by its estimate increment $\Delta\widehat{R}^{(c_n,t)}$.

$$\widehat{R}^{(c_n,t)} = R^{(c_n,t-1)} + \Delta\widehat{R}^{(c_n,t)} \tag{9}$$

The enhanced estimate increment $\Delta\widehat{R}^{(c_n,t)}$ is the evaluation of internal and external factors, which adds a growth factor λ compared to `OCEAN`.

$$\Delta\widehat{R}^{(c_n,t)} = \lambda \cdot \left(k_{in}^{(c_n)}\Delta\widehat{R}_{in}^{(c_n,t)} + k_{ex}^{(c_n)}\Delta\widehat{R}_{ex}^{(c_n,t)}\right), \quad k_{in}^{(c_n)}\&k_{ex}^{(c_n)} \sim \mathcal{N}(0.5, 0.1) \tag{10}$$

Growth factor of estimate incremental is used to reflect the impact of some socio-economic factors on reports growth, which obeys the following rule.

$$\begin{cases} \lambda = 1, & r < (b^{c_n})^{\tau/a^{(c_n)}} \\ \lambda \sim \mathcal{N}(0, 0.2), & r \geq (b^{c_n})^{\tau/a^{(c_n)}} \end{cases} \tag{11}$$

The enhanced estimate of intrinsic and extrinsic increment is calculated same as above.

$$\Delta\widehat{R}_{in}^{(c_n,t)} = \alpha \cdot \Delta R^{(c_n,t-1)}, \quad \alpha \sim \mathcal{N}(1.0, 0.2) \tag{12}$$

$$\Delta\widehat{R}_{ex}^{(c_n,t)} = \sum_{c_i \in \mathcal{NB}(c_n)} \beta_i \cdot S(c_n, c_i)\Delta R^{(c_i,t-1)}, \quad \beta_i \sim \mathcal{N}(0.3, 0.1) \tag{13}$$

6 Task III: Crisis Respond Strategies

6.1 Strategy Description

In our models, drug report increment of a county is determined by both intrinsic and extrinsic factors. For a local government, it is undoubtedly the most effective way to curb drug incidents from the source, internally and externally. For the intrinsic factors, we observe that it is highly correlated to veteran, disability, and marital status in **Task II**. For the extrinsic factor, the government can take measures to impede the spread of addictive drugs. Based on the analysis above, the government can take these strategies.

- For the counties where the detection of drugs exceeds the alert level, drug monitoring should be strengthened.
- For the disabled should receive more care from the government to prevent them from abusing drugs and their use of narcotic drugs should be limited. Giving more preferential treatment (e.g. creating job opportunities, setting up entrepreneurial projects) to the disabled is advocated to avoid them falling into addictive drugs.[13]

- The impact of veterans status on the amount of drug use is relatively large, and we infer that soldiers are not capable to adapt to ordinary life immediately after leaving the army or they might have a high rate of disability due to the war. Therefore, it is necessary to establish a complete veterans protection mechanism, such as updating veterans loan policies to help them get through the adjustment period quickly. [14]
- Divorced people have higher rates of opioid use. Medical institutions are supposed to pay close attention to the physical and mental states of these people to prevent them from abusing drugs.
- Control hospital use of narcotic drugs and increase the supervision of freight transport vehicles to impede the spread of drugs between neighboring counties.

These strategies embody some of the ideas of our model. Specifically, the external strategy impeding the spread of drugs between neighboring counties makes the extrinsic factor β smaller; the internal strategies decrease the use $a^{(c_n)}$ and trends-in-use $b^{(c_n)}$ factor. After adjusting the parameters and rerunning, the effectiveness of such strategies is proved by simulated results as illustrated in Section 6.2.

6.2 Quantify the Efficiency of the Strategy

In order to test the effectiveness of the strategy we proposed, we compared the simulated results before and after adjusting the parameters. We counted the number of the counties that awash in opioids before and after adjusting the parameters and use their ratio δ(higher is better) to quantify the efficiency of the proposed strategies. Strategy efficiency analysis is shown in **Table** 5.

7 Experiment Results

We used the Python to implement our algorithms. We process the provided data by neglecting the missing data. And the detail of the results are shown below.

7.1 Task I Results of `OCEAN`

Firstly, we use our model to obtain the possible origins of specific opioid in each state via performing reverse evolution on our cellular automata. For each specific opioid in each county, we simulate the drug report quantity in several years, and take the first year that the quantity is nonzero as the time that the specific opioid use started. Note that for the specific opioids that are first reported between 2010 and 2017, we simply take the year it first reported in the provided data as our result. For the drugs that never reported between 2010 and 2017, we assume that they won't start in recent years. Some simulated results are shown in Figure 3, and the complete results can be found in Appendix.

Secondly, we compute the total reported quantity of heroin and synthetic opioid between 2010 and 2017 and visualize the distribution(Figure 4). We observe that if the reported quantity of a specific drug is more than 200 and it kept increasing for more than 4 years, then the quantity tends to explode. Hence, the government should have specific concern on the counties that reach the drug identification threshold mentioned above.

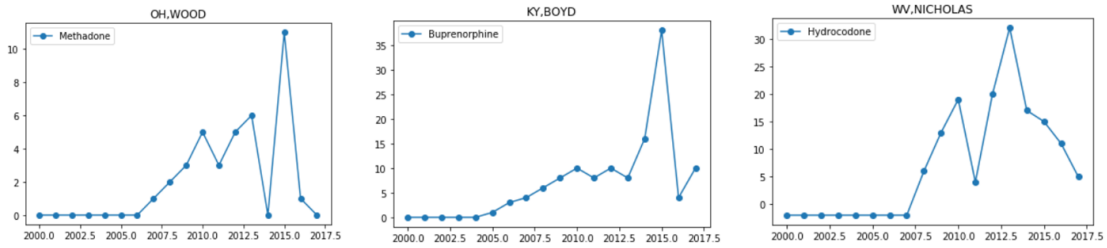

Figure 3: Reversed simulation results in **Task** I.

Figure 4: Distribution of heroin and synthetic opioid reports quantity

Thirdly, we simulate the reported quantity in 30 years for each specific drug. In our simulation, some of them tend to spread in some areas. The estimation of when and where they will reach the drug identification threshold levels are shown in Table 3.

7.2 Task II Results of D-OCEAN

In this task, we first compute the Pearson correlation coefficient between the reported quantity of the drugs and the factors in the provided U.S. Census socio-economic data. We observe that some factors like veteran, disability, and marital status are closely related to the reported quantity. The Pearson correlation coefficients are shown in Figure 5.

We choose the factors with the correlation coefficient higher than the threshold 0.85 as the main factors, then we apply K-means algorithm to cluster the counties into 4 classes and apply the corresponding score to them as described in Section 5.4. Figure, 6 shows an example of cluster result.

From Figure 6, we can clearly see that the counties are separated well. And although in most cases, the result of K-means algorithm may be different when the initial centers change, in our experiment, we run the K-means algorithm for several times but the result are the same. These indicate that the features we selected are distinguishable. After clustering, we compute the mean drug reported quantity inside each class, the results are shown in Table 4. We can see that the

Drug Name	County	Year
Buprenorphine	OH,HAMILTON	2018
Buprenorphine	PA,PHILADELPHIA	2019
Fentanyl	KY,BOONE	2014
Fentanyl	KY,CAMPBELL	2011
Fentanyl	PA,ALLEGHENY	2011
Fentanyl	PA,BEAVER	2019
Fentanyl	PA,MIFFLIN	2029
Fentanyl	PA,YORK	2011
Morphine	PA,PHILADELPHIA	2024
Oxycodone	OH,HAMILTON	2025
Oxycodone	OH,LAKE	2016
Oxycodone	OH,SUMMIT	2027
Oxycodone	OH,PHILADELPHIA	2010
Tramadol	OH,HAMILTON	2020
Tramadol	OH,MIAMI	2025
Tramadol	OH,MONTGOMERY	2021
Tramadol	PA,PHILADELPHIA	2016

Table 3: Breakouts prediction of opioids in the future

mean reported quantity varies greatly from one class to another, this further proof that the feature we selected are closely related to the reported quantity. Hence the use factor $a^{(c_n)}$ and trends-in-use factor $b^{(c_n)}$ can well characterize the use and trends-in-use pattern of a county. We then modify our model as described in Section 5.4 using the computed factors $a^{(c_n)}$ and $b^{(c_n)}$ to include some important factors in the dataset provided in **Task II**. Then we run our cellular automata again to obtain new simulated results. Some examples are showed in Figure 7 8.

Class	Mean quantity of the class
0	2284.7619
1	965.4268
2	290.9261
3	11762.6666

Table 4: Mean total drug reported quantity inside the cluster class.

It's obvious that the growth of the drug reported quantity in some counties slowed down. This illustrates that our `D-OCEAN` model performs better when take some important socio-economic factors into account.

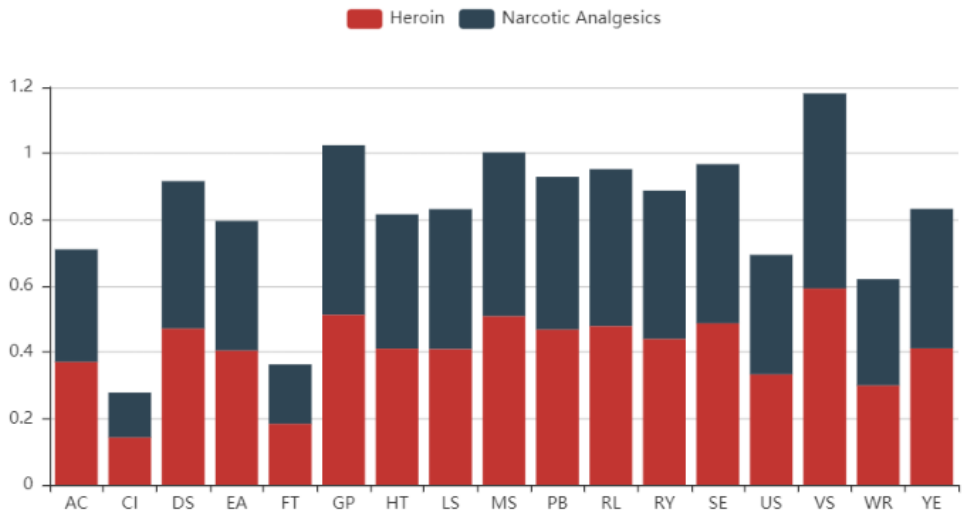

Figure 5: Pearson correlation coefficients: demographic features - report quantity.(The meaning of the abbreviations on the axis can be found in Appendix6.)

7.3 Task III. Crisis Respond Strategies

As mentioned in Section 6.2, we conduct our experiment using several different vaules for parameters $\beta, a^{(c_n)}, b^{(c_n)}$, and test the effectiveness by computing the number of counties awash in drugs under different parameter setting. The results are shown in **Table** 5.

β	**Upperbound of** $a^{(c_n)}$	**Lowerbound of** $b^{(c_n)}$	δ
$\beta \sim \mathcal{N}(0.3, 0.1)$	1.0	1.0	1.0
$\beta \sim \mathcal{N}(0.1, 0.1)$	1.0	1.0	1.4
$\beta \sim \mathcal{N}(0.3, 0.1)$	1.0	0.7	5.3
$\beta \sim \mathcal{N}(0.3, 0.1)$	0.7	1.0	3.2
$\beta \sim \mathcal{N}(0.3, 0.1)$	0.7	0.7	10.7

Table 5: Quantity results of the effectiveness of strategies in **Task III**. (Ratio δ - higher is better - is defined in Section 6.2.)

We can see that a small β and a small upperbound of $a^{(c_n)}, b^{(c_n)}$ can effectively tackle the opioid crisis, and in our experiment, we found that when $a^{(c_n)} < 0.75$ and $b^{(c_n)} < 0.85$, the proposed strategy is very effective to counter the opioid crisis.

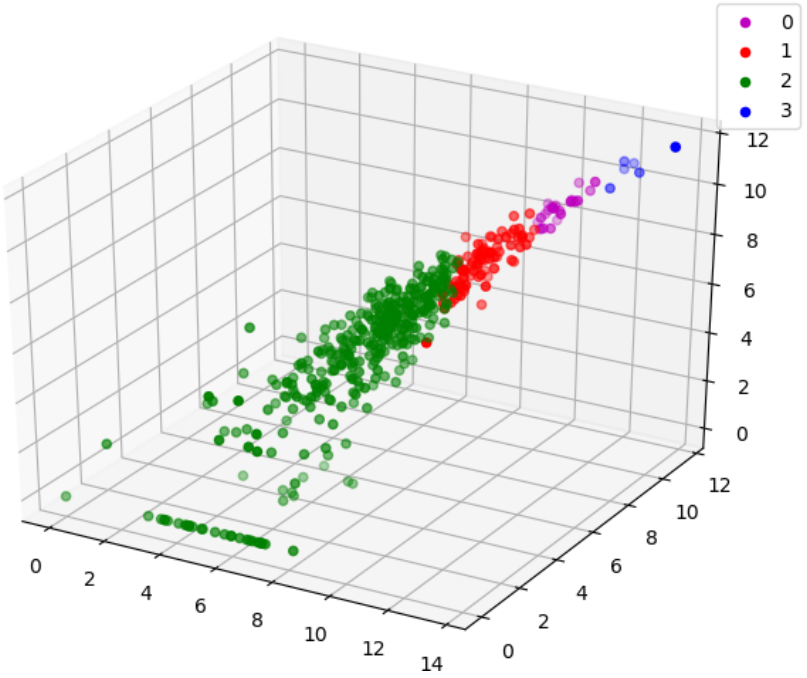

Figure 6: An example of K-means results.

8 Further Analysis

8.1 Effectiveness Analysis

In order to evaluate the effectiveness of our model, we use the provide data in **Task I** between year 2010 and 2013 to compute some parameters for our model, and then we run our cellular automata to simulate the reported quantity of each drug in each county between year 2014 and 2017, then we compare the simulated results with the given data. If the trend of reported quantity of a specific drug in a county in a specific year are same in the simulated results and the given data(e.g., the quantity increase in both the simulated results and the given data in that year), we consider our the prediction is right in that year. In this way, we compute our prediction accuracy and the results are shown in Figure 9. We can see that our prediction accuracy is high which confirms the effectiveness of our model.

8.2 Sensitivity Analysis

We test the sensitivity by adjusting different parameters in the model. Firstly, as shown in results of **Task II**, our K-means algorithm reach same result when different initial centers are selected, this indicates that K-means algorithm is not sensitive to the initial centers using the features we selected. Secondly, we decrease the threshold of the Pearson correlation coefficient, in this way, there are more socio-economic factors that we should consider, but the influence on the result is so slightly that it can be neglect. Finally, we apply different values to other parameters in our model, then we found that our model is sensitive to the α, β(incremental parameters), k_{in}, k_{ex} (weights of intrinsic and extrinsic opioid growth quantity) and λ(growth factor of estimate incremental). We

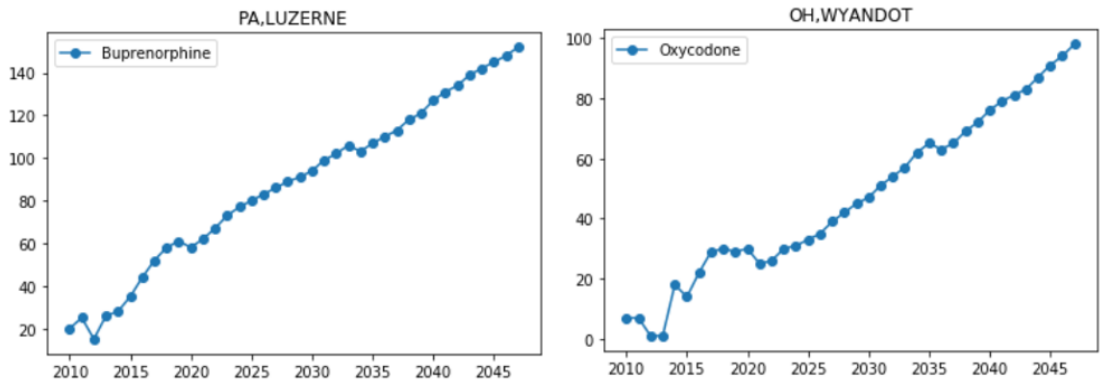

Figure 7: Simulation results: Buprenorphine in PA,LUZERNE and Oxycodone in OH,WYANDOT using `OCEAN` model.

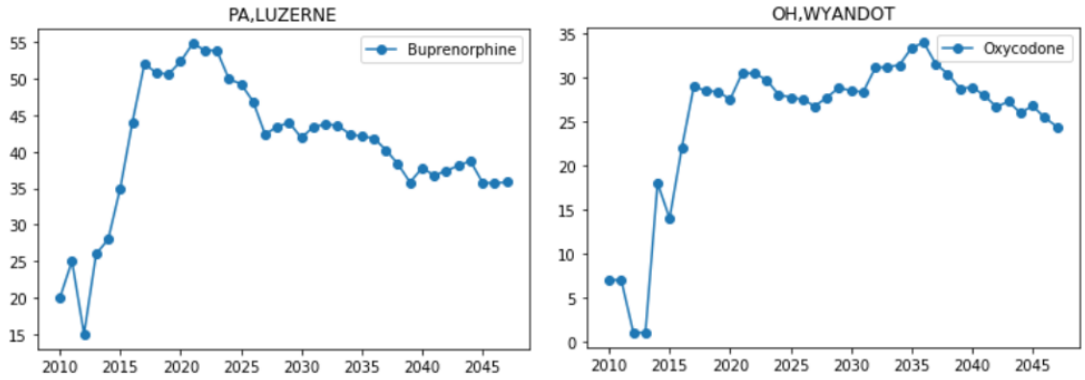

Figure 8: Simulated results of Buprenorphine in PA,LUZERNE and Oxycodone in OH,WYANDOT using `D-OCEAN` model.

argue that in real world, the drug reported quantity increasing pattern is also sensitive to those factors.

9 Strengths and Weaknesses

9.1 Strengths

- **High Scalability:** Although designed in detail, our model actually presents some macro discussions. Government can put our strategies into practice from many aspects in similar situations. We only introduce some typical features of drug incidents, and try to simulate the changing process by computers. As we do not fit problems into specific models, our models take effect in most cases.
- **High Reliability:** High reliability is ensured by the following two techniques.
  - ***Bilateral Analysis:*** In D-OCEAN, our results have been **reversely verified** to en-

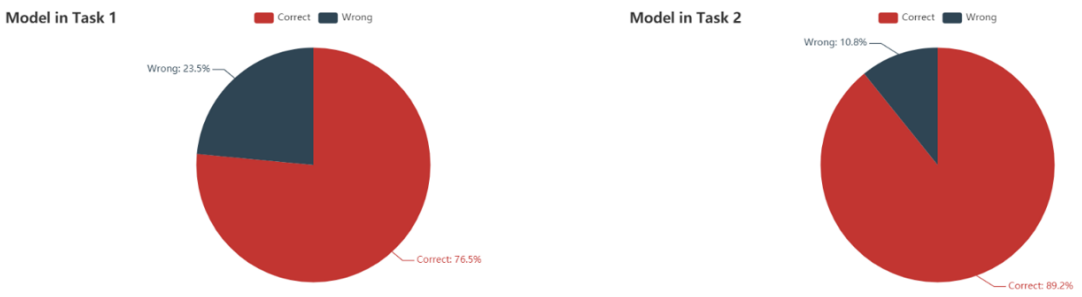

Figure 9: The prediction accuracy of our model.

sure reliability. We first conduct correlation calculation to screen out highly correlated demographic features. Luckily, results of this step are verified by the fact that points are separated well by K-means algorithm.

- ***Mass Data Analysis:*** We used large amounts of data in real dataset in latest years. In **Task I**, we deal with data with 63 kinds of opioids and 24062 drug reports. In **Task II**, 17 demographic features and 198 valid subordinate classes are involved to solve our problem. Strong data support is the guarantee of our reliability.

- **Low Cost:** Although the validity of our model is based on a large architecture, each part of the system is not of high complexity. **Cosine** and **Pearson** correlation analysis are both efficient similarity measurement methods. **K-means** algorithm only has linear time complexity. Core steps in our models are all economical but effective.

9.2 Weaknesses

- **Lack of Old Data**: The data we get are not very comprehensive, to some extent, affecting the accuracy of our results on relevant situations.

  - ***Incomplete Year:*** Data we collect are from latest years, but many opioid epidemics begin extremely early before 2000. Our study lacks consideration of the early conditions, which means that our estimate might not be so accurate for in early years.
  - *No Other States:* Limited by the data given, we have difficulties to have a clear clue about other states. This prevent us from depicting a overall opioid profile of the U.S.

- **Relatively Crude Consideration:** Due to time constraints, we are unable to draw a splendid picture of opioid cases without some slightly influencing factors (e.g. the production of certain opioid-related raw materials). Nevertheless, we considered the main factors in general cases and proved the validity of our model. Once these tiny factors are considered in detail, our model becomes more complete.

- **Lack Methods Comparison**: Due to time constraints, we only adapt the most advanced model as far as we known. There may exist other more desirable models to explore in the future.

10 Memo for Chief Administrator

Dear Sir:

As the opioid crisis intensified, we are beginning to realize the seriousness of the problem and all trying to find causes of this catastrophe. Upon hearing your program is trying to find associated information with complete database, our teams are honored to engage in investigating drug cases evolution and characterization. We have been working with painstaking efforts so as to make an accurate prediction, and we do have some outcome by researching day and night these days, which we hope can offer some important insights for you.

Inspired by the performance of cellular automaton in research of complex socio-economic systems, we employed it for simulating the evolution patterns of opioids. Taking account of demographic heterogeneity in each state, we enhanced our model by introducing some socio-economic factors. Finally, corresponding solutions were put forward, according to the analysis of our models. Our insights are listed as follows.

On one hand, the drugs might spread from one county to another. As the experimental results show, we find that geography is an important factor that affects the level of drug detection. When the drug detection level in a county rises, so does the detection level in the surrounding counties. Therefore, we suggest that when the detection level of some counties is high, the government need to strengthen the supervision on these counties to impede the spread of drug use.

On the other hand, we also analyzed the correlation between the county's socio-economic characteristics and the level of drug detection, and we found that they were highly relevant in some indicators like number of veterans, disabled. Our analysis indicates that the number of vulnerable groups in society such as widows, persons with disabilities, veterans, etc. was highly correlated with the number of drug addicts. We believe that targeted assistance to such groups will enrich their daily lives and improve their well-being. We believe that in this way, the number of people who choose to use drugs among these groups will be greatly reduced.

In conclusion, we recommend the U.S. government to take following strategies to cope with this severe problem. Internally, medical institutions should try to limit the amount of narcotics used by special groups (such as the disabled and retired veterans) and improve the existing medical security system; local governments are supposed to raise their oversight of vulnerable audiences with high rates of opioid addiction. Externally, strengthen drug control in counties, where drugs reports have exceeded the alert level, and their nearby.

With our validation, we think these actions can truly improve the opioid epidemic situation and our sensitivity analysis can tell you the utility of each strategy. We firmly believe that our efforts will create a better medical and living environment for all!

Best regards,

Sincerely

References

[1] DEA proposal will significantly cut opioid manufacturing in 2019. https://www.cdc.gov/features/confronting-opioids/index.html .

[2] Wilson M. Compton, Christopher M. Jones, and Grant T. Baldwin. Relationship between non-medical prescription-opioid use and heroin use. *New England Journal of Medicine*, 374(2):154–163, 2016. PMID: 26760086.

[3] Senate passes opioid crisis response act. https://www.aha.org/news/headline/2018-09-18-senate-passes-opioid-crisis-response-act .

[4] National Media Affairs Office. DEA proposal will significantly cut opioid manufacturing in 2019. https://www.dea.gov/press-releases/2018/08/16/justice-department-dea-propose-significant-opioid-manufacturing-reduction .

[5] Andrew Kolodny, David T. Courtwright, Catherine S. Hwang, Peter Kreiner, John L. Eadie, Thomas W. Clark, and G. Caleb Alexander. The prescription opioid and heroin crisis: A public health approach to an epidemic of addiction. *Annual Review of Public Health*, 36(1):559–574, 2015. PMID: 25581144.

[6] D R Mackintosh and G T Stewart. A mathematical model of a heroin epidemic: implications for control policies. *Journal of Epidemiology & Community Health*, 33(4):299–304, 1979.

[7] Emma White and Catherine Comiskey. Heroin epidemics, treatment and ode modelling. *Mathematical Biosciences*, 208(1):312 – 324, 2007.

[8] G. P. Samanta. Dynamic behaviour for a nonautonomous heroin epidemic model with time delay. *Journal of Applied Mathematics and Computing*, 35(1):161–178, Feb 2011.

[9] Adit A. Ginde, Mark C. Liu, and Jr Camargo, Carlos A. Demographic Differences and Trends of Vitamin D Insufficiency in the US Population, 1988-2004. *Archives of Internal Medicine*, 169(6):626–632, 03 2009.

[10] Jennifer A. Chatman and Francis J. Flynn. The influence of demographic heterogeneity on the emergence and consequences of cooperative norms in work teams. *Academy of Management Journal*, 44(5):956–974, 2001.

[11] Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644, Jul 1983.

[12] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[13] HARLAN HAHN. Introduction: Disability policy and the problem of discrimination. *American Behavioral Scientist*, 28(3):293–318, 1985.

[14] Gregg H. Gilbert, Laurence G. Branch, and Jeffrey Longmate. Dental care use by u.s. veterans eligible for va care. *Social Science & Medicine*, 36(3):361 – 370, 1993.

Appendices

The meaning of the abbreviations on the axis.

Abbreviation	Category
AC	ANCESTRY
CI	COMPUTERS AND INTERNET USE
DS	DISABILITY STATUS OF THE CIVILIAN NONINSTITUTIONALIZED POPULATION
EA	EDUCATIONAL ATTAINMENT
FT	FERTILIT
GP	GRANDPARENTS
HT	HOUSEHOLDS BY TYPE
LS	LANGUAGE SPOKEN AT HOME
MS	MARITAL STATUS
PB	PLACE OF BIRTH
RL	RELATIONSHIP
RY	RESIDENCE 1 YEAR AGO
SE	SCHOOL ENROLLMENT
US	U.S. CITIZENSHIP STATUS
VS	VETERAN STATUS
WR	WORLD REGION OF BIRTH OF FOREIGN BORN
YE	YEAR OF ENTRY

Table 6: Explanation of abbreviated words of category

State / DrugName	OH	WV	VA	KY	PA
MT-45	/	/	/	/	42081
Oxymorphone	39001, 39005, 39009	21015, 54039, 39107	21037, 51047, 51051	21009, 21015, 21029	42003, 42005, 42007
o-Fluorofentanyl	39057, 39071, 39101	/	51125, 51179	/	/
3-Fluorofentanyl	/	/	21073	/	/
3,4-Methylenedioxy U-47700	/	/	/	/	42003
U-49900	39007, 21073, 39063	/	/	/	/
Cyclopropyl/Crotonyl Fentanyl	39035, 39153	/	/	/	/
p-methoxybutyryl fentanyl	/	/	/	/	42003
Methorphan	39061, 39063, 39065	54003, 54057	51059	21067, 21111	21073
Oxycodone	21003, 39005, 39007	54003, 21015, 54011	51001, 51005, 51009	21005, 21009, 21011	39001, 42003, 42005
Fluoroisobutyryl fentanyl	39061	/	51041, 51059, 51069	39113, 21185	42013, 42015, 42021
Butyryl fentanyl	39061, 39173	39173	51003	21015, 21037, 21067	39111
Morphine	21003, 39007, 39009	54003, 54007, 54025	51001, 51005, 51015	21005, 21009, 21013	42003, 42005, 42007
Furanyl fentanyl	21073	/	51001, 51003, 51510	21019, 21043, 21049	42095
Opiates	39089	54001, 54003, 21015	/	/	39113
U-47700	39001, 39007, 39013	/	51013, 51041, 51059	21015, 21111, 21117	42003, 42007, 42011
Dihydromorphone	/	54001, 54003, 21015	/	/	39033
U-48800	21003, 39011, 21049	/	/	/	42003, 42079
Alphaprodine	/	/	/	/	39001, 42003, 42005
Butorphanol	39145	/	51810	/	39107
Cyclopentyl fentanyl	/	/	/	21067	/
Carfentanil	39001, 21003, 39005	54081	51059	21015, 21029, 21037	42003, 42007, 42021
Levorphanol	/	/	/	/	42003
Pethidine	39121	54001, 54003, 21015	51107	21111	21067, 42053, 21073
Hydromorphone	21003, 39017, 39035	54031, 54047, 39107	51015, 42009, 51033	21013, 21029, 21047	42003, 42017, 42021
4-Fluoroisobutyryl fentanyl	39057	54003, 54057	51510, 51041, 51059	21151, 39113	42003, 42011, 42013
Acetyldihydrocodeine	/	/	/	/	42101
Benzylfentanyl	39009, 39017, 21049	/	51059	/	42003
Nalbuphine	/	/	51041	/	/
p-Fluorobutyryl fentanyl	39173	/	51041	21151	42025, 42069, 42079
Methadone	21003, 39005, 39007	54039, 39173	51013, 51015, 42009	21005, 21009, 21011	39001, 42017, 42025
trans-3-Methylfentanyl	/	/	/	21043	42003
Methoxyacetyl fentanyl	39005, 39017, 21041	/	51003, 51075, 51085	21015, 21037, 21049	39001, 42003, 42005

State / DrugName	OH	WV	VA	KY	PA
Valeryl fentanyl	39077	/	/	/	42003, 42011, 42013
Remifentanil	/	/	/	/	42003
Cyclopropyl fentanyl	39001, 21003, 39005	/	51550, 51041, 51047	21227	42003, 42007, 42037
4-Methylfentanyl	39095	/	/	/	/
3-Methylfentanyl	21003, 39007, 39013	54003	51059, 51061, 51069	/	42007, 42011, 39017
ANPP	39113	54081	51059, 51153, 51179	39113	42043
Fentanyl	21003, 39013, 39035	54051	51059, 51087, 51107	21059, 21111, 21141	42003, 42013, 42017
Fluorobutyryl fentanyl	39151	/	/	/	/
Tetrahydrofuran fentanyl	21049, 21151	/	/	/	42133
Meperidine	39151	54001, 54003, 21015	51095	21009	42101
Buprenorphine	21003, 39007, 39009	54003, 54007, 54011	51023, 51027, 21041	21005, 21009, 21043	42003, 42005, 42007
Hydrocodeinone	/	/	/	/	42077, 42097
Phenyl fentanyl	39029, 39035, 39041	/	/	/	42003
p-Fluorofentanyl	39035	/	/	/	/
Desmethylprodine	39113, 39149, 21227	/	51169	21111	/
Dihydrocodeine	39005	54001, 54003, 21015	51041, 51153	21073	42133
Furanyl/3-Furanyl fentanyl	/	/	/	/	/
Opium	39133	54001, 54003, 21015	51059	21117	42101
Metazocine	/	/	/	/	42101
Crotonyl fentanyl	39017, 21049, 39035	/	/	/	/
Dextropropoxyphene	39149	54039, 39107	/	/	42107
Fluorofentanyl	39113	/	/	21085	/
Mitragynine	39035, 39113	/	51710	21117	42011, 42077, 42107
U-VA7WV	39035	/	/	/	/
Isobutyryl fentanyl	39061	/	/	/	/
cis-3-methylfentanyl	/	/	/	21043	42003
Codeine	21003, 39007, 39025	54039	51013, 51027, 51041	21073, 21089, 39065	42003, 42013, 42017
Acetylcodeine	/	54069, 21191, 54073	51059	/	42101, 21195, 42105
Acryl fentanyl	39005, 21049, 39035	54057	51171	21217	42133
Pentazocine	21073, 39153	54059	51061, 51197	21111	39041
Thebaine	39017	/	/	/	21195
Acetyl fentanyl	39101, 39113, 39149	54003, 54033, 54039	51041, 51153	21019, 21023, 21043	42047, 39057, 42101
Hydrocodone	21003, 39005, 39009	54003, 21015, 54011	51001, 51005, 51011	21001, 21007, 21009	42003, 42009, 42011

Code

Part I - **corrcompute.py**

```
1
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import numpy as np
8  import pandas as pd
9
10 data = pd.read_csv('MCM_NFLIS_Data.csv')
11
12 # data
13 # RangeIndex: 24062 entries, 0 to 24061
14 # Data columns(total 10 columns):
15 # YYYY                    24062 non-null int64
16 # State                   24062 non-null object
17 # COUNTY                  24062 non-null object
18 # FIPS_State              24062 non-null int64
19 # FIPS_County             24062 non-null int64
20 # FIPS_Combined           24062 non-null int64
21 # SubstanceName           24062 non-null object
22 # DrugReports             24062 non-null int64
23 # TotalDrugReportsCounty  24062 non-null int64
24 # TotalDrugReportsState   24062 non-null int64
25 # dtypes: int64(7), object(3)
26 #
27 import scipy.stats as stats
28 import matplotlib.pyplot as plt
29 state = sorted(list(set(data['State'])))
30 year = sorted(list(set(data['YYYY'])))
31
32 fips = sorted(list(set(data['FIPS_Combined'])))
33 he_list = []
34 na_list = []
35 for i in range(len(fips)):
36     f = fips[i]
37     if i % 20 == 0:
38         print(i)
39     y_he = []
40     y_na = []
41     for y in year:
42         y_he.append(np.sum(data['DrugReports'][(data.SubstanceName=='Heroin')\
43          & (data.FIPS_Combined==f) & (data.YYYY==y)]))
44         y_na.append(np.sum(data['DrugReports'][(data.SubstanceName!='Heroin')\
45          & (data.FIPS_Combined==f) & (data.YYYY==y)]))
46     y_he = np.array(y_he)
```

```
47      y_na = np.array(y_na)
48      he_list.append(y_he)
49      na_list.append(y_na)
50
51  print('done first')
52
53
54  # In[171]:
55
56
57  def relate(a, b):
58      #c = a / np.max(a)
59      #d = b / np.max(b)
60      #return np.linalg.norm(c-d)
61      a_sub = a[1:]-a[:-1]
62      b_sub = b[1:]-b[:-1]
63      fenzi = a_sub @ b_sub
64      fenmu = np.linalg.norm(a_sub) * np.linalg.norm(b_sub)
65      return fenzi / fenmu
66
67  def fips_to_name(data, a):
68      name = data['State'][data.FIPS_Combined==a]
69      name = list(name)[0]
70      name += ',' + list(data['COUNTY'][data.FIPS_Combined==a])[0]
71      return name
72  count = 0
73  for i in range(len(na_list)):
74      for j in range(len(na_list)):
75          if i != j:
76              #if np.sum(na_list[i]) > 5 and np.sum(na_list[j]) > 5:
77              re = relate(na_list[i], na_list[j])
78              if re > 0.97:
79                  plt.plot(year, na_list[i], marker='o', \
80                      label=fips_to_name(data,fips[i]))
81                  plt.plot(year, na_list[j], marker='x', \
82                      label=fips_to_name(data,fips[j]))
83                  plt.legend()
84                  plt.title(str(re))
85                  plt.show()
86                  count+=1
87  print(count)
88
89
90  # In[114]:
91
92
93  state = sorted(list(set(data.State)))
94  drug = sorted(list(set(data.SubstanceName)))
95  drug.remove('Heroin')
```

```
count = 0
#result_file = open('start_county.txt', 'w')
to_deal = []
for s in state:
    #for d in drug:
    for d in sorted(list(set(data['SubstanceName'][data.State==s]))):
        drug_trend = []
        for y in year:
            drug_trend.append(np.sum(data['DrugReports'][(data.State==s) \
                & (data.YYYY==y) & (data.SubstanceName==d)]))

        '''
        if drug_trend[0] == 0:
            county = sorted(list(set(data['COUNTY'][data.State==s])))
            tmp = np.array(drug_trend)
            if np.sum(tmp) == 0:
                print(s+','+d+': Never reported.')
                #content = s+','+d+': Never reported.\n'
                #result_file.write(content)
            else:
                first_appear = []
                y = np.array(year)[np.greater(tmp, 0)][0]
                for c in county:
                    if np.sum(data['DrugReports'][(data.State==s)\
                     & (data.YYYY==y) & (data.SubstanceName==d)\
                      & (data.COUNTY==c)]) > 0:
                        first_appear.append(c)
                #content = s+','+d+' first reported in '+str(y)+' in '\
                +str(first_appear)+'.\n'
                #result_file.write(content)
            #print(str(s)+','+str(d))
            #plt.plot(year, drug_trend, label=fips_to_name(data,fips[i]))
            #plt.legend()
            #plt.title(str(s)+','+str(d))
            #plt.show()
        else:
        '''

        if drug_trend[0] > 0:
            to_deal.append((s,d))
print(to_deal)
tmpp = []
for deal in to_deal:
    tmpp.append(deal[1])
tmpp = set(tmpp)
print(tmpp)


# In[70]:
```

```
145
146
147 count = 0
148 relations = []
149 for i in range(len(na_list)):
150     for j in range(len(na_list)):
151         if i != j:
152             #if np.sum(na_list[i]) > 5 and np.sum(na_list[j]) > 5:
153             re = relate(na_list[i], na_list[j])
154             relations.append(re)
155         else:
156             relations.append(0.0)
157 print(len(relations))
158
159
160 # In[73]:
161
162
163 relations = np.array(relations)
164 relations = np.reshape(relations, [len(fips), len(fips)])
165 np.save('relations.npy', relations)
166
167
168 # In[75]:
169
170
171 a = np.load('relations.npy')
172 print(a)
173
174
175 # In[155]:
176
177
178 coordinates = np.load('../data_processed/new_coor.npy')
179 distances = []
180 for i in range(coordinates.shape[0]):
181     for j in range(coordinates.shape[0]):
182         distances.append(np.linalg.norm(coordinates[i]-coordinates[j]))
183 distances = np.array(distances)
184 distances = np.reshape(distances, [coordinates.shape[0], coordinates.shape[0]])
185 print(distances)
186
187
188 # In[164]:
189
190
191 threshold = np.mean(np.sort(distances)[:,10])
192 neighbor = np.less(distances, threshold).astype(np.int32)
193 neighbor -= np.eye(neighbor.shape[0], dtype=np.int32)
```

```
#np.save('neighbor.npy', neighbor)
print(np.where(np.greater(np.sort(distances)[:,1],2)))


# In[166]:


need_deal = tmpp
print(need_deal)
for d in need_deal:
    init_increase = []
    init_report = []
    y2 = year[-1]
    y1 = year[-2]
    for f in fips:
        report1 = np.sum(data['DrugReports'][(data.FIPS_Combined==f) \
            & (data.SubstanceName==d) & (data.YYYY==y1)])
        report2 = np.sum(data['DrugReports'][(data.FIPS_Combined==f) \
            & (data.SubstanceName==d) & (data.YYYY==y2)])
        if len(list(data['DrugReports'][(data.FIPS_Combined==f) \
            & (data.SubstanceName==d) & (data.YYYY==y1)])) > 1:
            print(error)
        init_increase.append(report2-report1)
        init_report.append(report2)
    init_increase = np.array(init_increase)
    init_report = np.array(init_report)
    np.save('data/pos/'+d+'_init_increase.npy', init_increase)
    np.save('data/pos/'+d+'_init_report.npy', init_report)


# In[167]:


deal_file = open('substances_to_deal.txt', 'w')
for d in need_deal:
    deal_file.write(d+' ')
```

Part I - **prob1.py**

```
import numpy as np
import pandas as pd
import random
import os

class county(object):
    def __init__(self, fips, init_self_increase, report_num):
        self.fips = fips
        self.last_increase = init_self_increase
        self.report_num = report_num

```

```
12      def _get_self_increase(self):
13          # linear increasing in a county itself
14          mu = 1.0
15          sigma = 0.2
16          increase = random.normalvariate(mu, sigma) * self.last_increase
17          return increase
18
19      def update(self, neighbor_influence):
20          # consist of two parts: self increase and neighbor influence
21          # return total increase
22          if self.report_num <= 0:
23              return 0
24
25          self_increase = self._get_self_increase()
26          self_coeff = random.normalvariate(0.5, 0.1)
27          neighbor_coeff = 1-self_coeff
28          total_increase = \
29          np.rint(self_increase*self_coeff + neighbor_influence*neighbor_coeff)
30          if random.random() < 0.1:
31              self.report_num -= total_increase
32          else:
33              self.report_num += total_increase
34          return total_increase
35
36  def _get_neighbor_influence(last_increases, neighbors, relation):
37      # return a dict: fips: neighbor increase
38      all_fips = list(last_increases.keys())
39      neighbor_influence = dict()
40      for fips in all_fips:
41          #coeff*relation
42          influence = 0.0
43          neighbor_fips = neighbors[fips]
44          for n_fips in neighbor_fips:
45              coeff = random.normalvariate(0.3,0.1)
46              influence += \
47              relation[fips][n_fips]*coeff*last_increases[n_fips]
48          influence = np.rint(influence)
49          neighbor_influence[fips] = influence
50      return neighbor_influence
51
52  def simulate(init_increases, init_reports, neighbors, relation):
53      # init_increases: a dict: fips to init_increases
54      counties = []
55      last_increase = init_increases.copy()
56      all_fips = list(init_increases.keys())
57      for fips in all_fips:
58          counties.append(county(fips, init_increases[fips], init_reports[fips]))
59      simulate_years = 30
60      results = [] # each entry is the result of a year, stored in a dict
```

```
61      for _ in range(simulate_years):
62          # simulate one year
63          neighbor_influence = \
64          _get_neighbor_influence(last_increase, neighbors, relation)
65          result = dict()
66          for i in range(len(counties)):
67              fips = counties[i].fips
68              increase = counties[i].update(neighbor_influence[fips])
69              last_increase[fips] = increase
70              result[fips] = counties[i].report_num
71          results.append(result)
72      return results
73
74  def load_data(substance_name):
75      init_increases = dict()
76      init_reports = dict()
77      neighbor = dict()
78      relations = dict()
79      increase_data = np.load('data/pos/'+substance_name+'_init_increase.npy')
80      report_data = np.load('data/pos/'+substance_name+'_init_report.npy')
81      relation_data = np.load('relations.npy')
82      neighbor_data = np.load('neighbor.npy')
83      all_fips = _get_fips()
84      for i in range(len(all_fips)):
85          fips = all_fips[i]
86          init_increases[fips] = increase_data[i]
87          init_reports[fips] = report_data[i]
88          dict_tmp = dict()
89          list_tmp = []
90          for j in range(len(all_fips)):
91              dict_tmp[all_fips[j]] = relation_data[i][j]
92              if neighbor_data[i][j] == 1:
93                  list_tmp.append(all_fips[j])
94          relations[fips] = dict_tmp
95          neighbor[fips] = list_tmp
96      return init_increases, init_reports, neighbor, relations
97
98
99  def write_result(substacne_name, results):
100     all_fips = sorted(list(results[0].keys()))
101     result_npy = []
102     for fips in all_fips:
103         for i in range(len(results)):
104             result_npy.append(results[i][fips])
105     result_npy = np.array(result_npy)
106     result_npy = np.reshape(result_npy, [len(all_fips), -1])
107     if not os.path.exists('results/pos'):
108         os.makedirs('results/pos')
109     np.save('results/pos/'+substacne_name+'.npy' , result_npy)
```

```
110
111 def _get_fips():
112     data = pd.read_csv('MCM_NFLIS_Data.csv')
113     fips = sorted(list(set(data['FIPS_Combined'])))
114     return fips
115
116 def _get_substances():
117     substance_file = open('substances_to_deal.txt')
118     for line in substance_file:
119         substances = line.split(' ')
120     return substances
121
122 if __name__ == '__main__':
123     substances = _get_substances()[:-1]
124     for substance in substances:
125         init_increases, init_reports, \
126             neighbors, relation = load_data(substance)
127         results = simulate(init_increases, init_reports, neighbors, relation)
128         write_result(substance, results)
129         print(substance)
130     #test()
```

Part II - **prob2.py**

```
1  import numpy as np
2  import pandas as pd
3  import random
4  import os
5
6  class county(object):
7      def __init__(self, fips, init_self_increase, \
8          report_num, increase_count, b_value, a_value):
9          self.fips = fips
10         self.last_increase = init_self_increase
11         self.report_num = report_num
12         self.increase_count = increase_count
13         self.b_value = b_value
14         self.a_value = a_value
15
16     def _get_self_increase(self):
17         # linear increasing in a county itself
18         mu = 1.0
19         sigma = 0.2
20         increase = random.normalvariate(mu, sigma) * self.last_increase
21         return increase
22
23     def update(self, neighbor_influence):
24         # consist of two parts: self increase and neighbor influence
25         # return total increase
26         if self.report_num <= 0:
```

```
27              return 0
28          self_increase = self._get_self_increase()
29          self_coeff = random.normalvariate(0.5, 0.1)
30          neighbor_coeff = 1-self_coeff
31          total_increase = np.rint(self_increase*self_coeff + \
32                                   neighbor_influence*neighbor_coeff)
33          #if random.random() < 0.1:
34          #    self.report_num -= total_increase
35          #else:
36          #    self.report_num += total_increase
37          if random.random() < self.b_value**(self.increase_count/self.a_value):
38              self.report_num += total_increase
39          else:
40              total_increase = random.normalvariate(0.0,0.4) * total_increase
41              self.report_num -= total_increase
42          if total_increase > 0:
43              self.increase_count += 1
44          else:
45              self.increase_count -= 0
46          return total_increase
47
48  def _get_neighbor_influence(last_increases, neighbors, relation):
49      # return a dict: fips: neighbor increase
50      all_fips = list(last_increases.keys())
51      neighbor_influence = dict()
52      for fips in all_fips:
53          #coeff*relation
54          influence = 0.0
55          neighbor_fips = neighbors[fips]
56          for n_fips in neighbor_fips:
57              coeff = random.normalvariate(0.3,0.1)
58              influence += relation[fips][n_fips]*coeff*last_increases[n_fips]
59          influence = np.rint(influence)
60          neighbor_influence[fips] = influence
61      return neighbor_influence
62
63  def simulate(init_increases, init_reports, neighbors, \
64      relation, increase_count, a_value, b_value):
65      # init_increases: a dict: fips to init_increases
66      counties = []
67      last_increase = init_increases.copy()
68      all_fips = list(init_increases.keys())
69      for fips in all_fips:
70          counties.append(county(fips, init_increases[fips], init_reports[fips],
71                                 increase_count[fips], a_value[fips], b_value[fips]))
72      simulate_years = 30
73      results = [] # each entry is the result of a year, stored in a dict
74      for _ in range(simulate_years):
75          # simulate one year
```

```
76          neighbor_influence = _get_neighbor_influence(last_increase, \
77              neighbors, relation)
78          result = dict()
79          for i in range(len(counties)):
80              fips = counties[i].fips
81              increase = counties[i].update(neighbor_influence[fips])
82              last_increase[fips] = increase
83              result[fips] = counties[i].report_num
84          results.append(result)
85      return results
86
87  def load_data(substance_name):
88      init_increases = dict()
89      init_reports = dict()
90      neighbor = dict()
91      relations = dict()
92      increase_count = dict()
93      a_value = dict()
94      b_value = dict()
95      increase_data = np.load('data/pos/'+substance_name+'_init_increase.npy')
96      report_data = np.load('data/pos/'+substance_name+'_init_report.npy')
97      relation_data = np.load('relations.npy')
98      neighbor_data = np.load('neighbor.npy')
99      increase_count_data = np.load('increase_count.npy')
100     a_value_data = np.load('a_value.npy')
101     b_value_data = np.load('b_value.npy')
102     all_fips = _get_fips()
103     for i in range(len(all_fips)):
104         fips = all_fips[i]
105         init_increases[fips] = increase_data[i]
106         init_reports[fips] = report_data[i]
107         increase_count[fips] = increase_count_data[i]
108         a_value[fips] = a_value_data[i]
109         b_value[fips] = b_value_data[i]
110         dict_tmp = dict()
111         list_tmp = []
112         for j in range(len(all_fips)):
113             dict_tmp[all_fips[j]] = relation_data[i][j]
114             if neighbor_data[i][j] == 1:
115                 list_tmp.append(all_fips[j])
116         relations[fips] = dict_tmp
117         neighbor[fips] = list_tmp
118     return init_increases, init_reports, neighbor, relations, \
119     increase_count, a_value, b_value
120
121
122 def write_result(substacne_name, results):
123     all_fips = sorted(list(results[0].keys()))
124     result_npy = []
```

```
125     for fips in all_fips:
126         for i in range(len(results)):
127             result_npy.append(results[i][fips])
128     result_npy = np.array(result_npy)
129     result_npy = np.reshape(result_npy, [len(all_fips), -1])
130     if not os.path.exists('results/prob2/pos/'):
131         os.makedirs('results/prob2/pos')
132     np.save('results/prob2/pos/'+substacne_name+'.npy' , result_npy)
133 
134 def _get_fips():
135     data = pd.read_csv('MCM_NFLIS_Data.csv')
136     fips = sorted(list(set(data['FIPS_Combined'])))
137     return fips
138 
139 def _get_substances():
140     substance_file = open('substances_to_deal.txt')
141     for line in substance_file:
142         substances = line.split(' ')
143     return substances
144 
145 if __name__ == '__main__':
146     substances = _get_substances()[:-1]
147     for substance in substances:
148         init_increases, init_reports, neighbors, relation, \
149         increase_count, a_value, b_value = load_data(substance)
150         results = simulate(init_increases, init_reports, \
151             neighbors, relation, increase_count, a_value, b_value)
152         write_result(substance, results)
153         print(substance)
154     #test()
```

Part II - **kmeans.py**

```
1 
2 # coding: utf-8
3 
4 # In[47]:
5 
6 
7 import pandas as pd
8 import numpy as np
9 
10 
11 # In[48]:
12 
13 
14 def compute_corr(data1, data2, name1, name2):
15     if type(data2[name2][0]) == type('ate'):
16         return 0.0
17     data = pd.merge(data1, data2, on='FIPS_Combined')
```

```
18      tmp1 = data[name1]
19      tmp2 = data[name2]
20      corr = tmp1.corr(tmp2, method='pearson')
21      if corr != corr:
22          return 0.0
23      return corr
24
25
26  # In[49]:
27
28
29  def trans(name, y):
30      if y < 2013:
31          t = pd.read_csv('tmp.csv')
32      else:
33          t = pd.read_csv('tmp1.csv')
34      t.columns = ['a','b']
35      return list(t['b'][t.a==name])[0]
36
37
38  # In[50]:
39
40
41  factors = set()
42  factor_raws = set()
43  report_data = pd.read_csv('trend_use.csv')
44  for y in range(2010,2017):
45      print('year:',y)
46      data2 = pd.read_csv('new_'+str(y)[2:]+'.csv')
47      cols = list(data2.columns)
48      for i in range(1, len(cols)):
49          tmp_data = data2[[cols[0],cols[i]]]
50          corr = compute_corr(report_data, tmp_data, 'NA_trend', cols[i])
51          #print(corr)
52          if np.abs(corr) > 0.75:
53              info = trans(cols[i], y)
54              #print(info, corr)
55              factors.add(info[info.find(';')+2:info.find('-')-1])
56              factor_raws.add((cols[i],y))
57  print(factors)
58  print(factor_raws)
59
60
61  # In[22]:
62
63
64  import random
65  def kmeans(data, k, threhold):
66      # random select k initial points
```

```
    centers = []
    for i in range(k):
        index = random.randint(0, data.shape[0]-1)
        centers.append(data[index])
    count = 0
    while(True):
        l2_distances = []
        for i in range(k):
            l2_distance = np.linalg.norm(data-centers[i], axis=1)
            l2_distances.append(np.expand_dims(l2_distance,axis=-1))
        l2_distances = np.concatenate(l2_distances, axis=1)
        classify_result = np.argmin(l2_distances, axis=1)
        error = 0.0
        for i in range(k):
            class_data = data[classify_result==i]
            if class_data.shape[0] != 0:
                new_center = np.mean(class_data, axis=0)
                error += np.linalg.norm(centers[i] - new_center)
                centers[i] = new_center
        count += 1
        #print("Iteration:%d, Error:%f" %(count, error))
        if error < threhold:
            break
    #print(centers)
    return classify_result


# In[23]:


import numpy as np
import pandas as pd
tmp_data = pd.read_csv('MCM_NFLIS_Data.csv')
all_fips = sorted(list(set(tmp_data['FIPS_Combined'])))
class_features = []
for big_class in factors:
    features = []
    #count=0
    for factor_raw in factor_raws:
        info = trans(factor_raw[0], factor_raw[1])
        info = info[info.find(';')+2:info.find('-')-1]
        if info == big_class:
            #count+=1
            raw_data = pd.read_csv('new_'+str(factor_raw[1])[2:]+'.csv')
            tmp_list = []
            for fips in all_fips:
                tmp_list.append(np.array(raw_data[factor_raw[0]]\
                    [raw_data.FIPS_Combined == fips]))
            tmp_list = np.concatenate(tmp_list)
```

```
            #print(tmp_list.shape)
            if tmp_list.shape[0] == 461:
                features.append(tmp_list)
    features = np.stack(features).T
    print(features.shape)
    class_features.append(features)


# In[24]:


report_num_data = pd.read_csv('trend_use.csv')
report_num_np = []
for fips in all_fips:
    report_num_np.append(np.sum(report_num_data['NA_trend']\
        [report_num_data.FIPS_Combined==fips]))
report_num_np = np.array(report_num_np)


# In[46]:


class_num = 5
#for i in range(class_num):
b_value = np.zeros(report_num_np.shape)
for feature_ind in range(len(class_features)):
    classify = kmeans(class_features[feature_ind], class_num, 1e-7)
    scores = []
    for i in range(class_num):
        class_score = np.mean(report_num_np[classify==i])
        print(class_score)
        scores.append(class_score)
    arg_index = np.argsort(np.array(scores))
    for i in range(class_num):
        b_value[classify==i] += arg_index[i]
print(b_value)
np.save('b_value.npy', b_value/np.max(b_value))
```

Part III - **prob3.py**

```

# coding: utf-8

# In[6]:


import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('../MCM_NFLIS_Data.csv')
all_fips = sorted(list(set(data['FIPS_Combined'])))
files = os.listdir('./')
files = [f for f in files if f[-4:] == '.npy']
print(files)


# In[7]:


def fips_to_name(data, a):
    name = data['State'][data.FIPS_Combined==a]
    name = list(name)[0]
    name += ',' + list(data['COUNTY'][data.FIPS_Combined==a])[0]
    return name


# In[37]:


year = list(range(2000,2018))
for ind in range(len(files)):
    result = np.load(files[ind])
    for i in range(len(all_fips)):
        if np.sum(np.abs(result[i])) > 0:
            to_plot = list(result[i])
            to_plot.reverse()
            ori = []
            d = files[ind][:-4]
            f = all_fips[i]
            for y in list(range(2010,2018)):
                ori.append(np.sum(data['DrugReports'][(data.SubstanceName==d) \
                    & (data.FIPS_Combined==f) & (data.YYYY==y)]))
            to_plot = to_plot + ori
            #print(to_plot)
            plt.plot(year, to_plot, marker='o', label=d)
            plt.legend()
            plt.title(fips_to_name(data, f))
            plt.show()


# In[79]:


year = list(range(2010,2048))
for ind in range(len(files)):
    result = np.load('pos/'+files[ind])
    for i in range(len(all_fips)):
        if np.max(np.abs(result[i])) > 20:
```

```
            to_plot = list(result[i])
            ori = []
            d = files[ind][:-4]
            f = all_fips[i]
            for y in list(range(2010,2018)):
                ori.append(np.sum(data['DrugReports'][(data.SubstanceName==d) \
                    & (data.FIPS_Combined==f) & (data.YYYY==y)]))
            to_plot = ori + to_plot
            #print(to_plot)
            plt.plot(year, to_plot, marker='o', label=d)
            plt.legend()
            plt.title(fips_to_name(data, f))
            plt.show()


# In[78]:


year = list(range(2010,2048))
for ind in range(len(files)):
    result = np.load('prob2/pos/'+files[ind])
    for i in range(len(all_fips)):
        if np.max(np.abs(result[i])) > 20:
            to_plot = list(result[i])
            ori = []
            d = files[ind][:-4]
            f = all_fips[i]
            for y in list(range(2010,2018)):
                ori.append(np.sum(data['DrugReports'][(data.SubstanceName==d) \
                    & (data.FIPS_Combined==f) & (data.YYYY==y)]))
            to_plot = ori + to_plot
            #print(to_plot)
            plt.plot(year, to_plot, marker='o', label=d)
            plt.legend()
            plt.title(fips_to_name(data, f))
            plt.show()


# In[63]:


states = [51,39,42,21,54]
file_write = open('start_county1.txt', 'w')
for ind in range(len(files)):
    result = np.load(files[ind])
    for state in states:
        first_year = 2010
        county_list = []
        for j in range(10):
```

```
            for i in range(len(all_fips)):
                if all_fips[i] // 1000 == state:
                    if j == 0 and result[i][j] <= 0:
                        first_year = 2009
                        county_list.append(fips_to_name(data, all_fips[i]))
                    elif j > 0 and j < 9:
                        if result[i][j] > 0 and result[i][j+1] <= 0:
                            if first_year != 2009 - j:
                                first_year = 2009 - j
                                county_list = [fips_to_name(data, all_fips[i])]
                            else:
                                county_list.append(fips_to_name(data, all_fips[i]))
                    else:
                        if result[i][j] > 0:
                            if first_year != 1998:
                                first_year = 1998
                                county_list = [fips_to_name(data, all_fips[i])]
                            else:
                                county_list.append(fips_to_name(data, all_fips[i]))

        file_write.write(str(state)+','+files[ind][:-4]+' first reported in '\
            +str(first_year)+' in '+str(county_list)+'\n')


# In[86]:


year = list(range(2010,2038))
count = 0
for ind in range(len(files)):
    result = np.load('pos/'+files[ind])[:,:20]
    for i in range(len(all_fips)):
        #if np.sum(np.abs(result[i])) > 20:
        if np.max(np.abs(result[i])) > 500:
            to_plot = list(result[i])
            ori = []
            d = files[ind][:-4]
            f = all_fips[i]
            for y in list(range(2010,2018)):
                ori.append(np.sum(data['DrugReports'][(data.SubstanceName==d) \
                    & (data.FIPS_Combined==f) & (data.YYYY==y)]))
            to_plot = ori + to_plot
            #print(to_plot)
            #plt.plot(year, to_plot, marker='o', label=d)
            #plt.legend()
            #plt.title(fips_to_name(data, f))
            #plt.show()
            print(d, fips_to_name(data, f), np.where(result[i]>500)[0][0]+2010)
print(count)
```