For office use only	Team Control Number	For office use only
T1 _____	**1920446**	F1 _____
T2 _____		F2 _____
T3 _____	Problem Chosen	F3 _____
T4 _____	**C**	F4 _____

2019
MCM/ICM
Summary Sheet

SOS: Spreading Model, Optimization, and Strategy Program for the Opioid Crisis

Summary

As the deteriorating Opioid Crisis overwhelms the United States, the DEA/National Forensic Laboratory Information System (NFLIS) are striving to control the nationwide spread of opioid overdose indidents. In this paper, we formuate a concrete and novel framework, named **SOS (Spreading Model, Optimization, and Strategy Program for the Opioid Crisis)**, seeking for new strategies to combat the exacerbating opioid crisis.

Based on the report provided, we create a Spreading Model of opioid incidents, focusing on the individual counties located in five U.S. states: Ohio (OH), Kentucky (KY), West Virginia (WV), Virginia (VA), and Pennsylvania (PA). We first get the longitude and latitude of every county and utilize a **Back-Propagation Neutral Network (BPNN)** to characterize the reported opioid and heroin incidents in and between the five states and their counties from 2010 to 2017. Based on this initial model, we next adopt **Outlier Detection** to identify some most possible locations where specific opioid accidents might have started, and the trending opioids crisis in the future of the five states.

Furthermore, we combine **Principal Component Analysis (PCA)** and **Deep Learning Approach Auto-encoder (AE)** techniques to analyze the U.S. Census Socio-economic data provided, aiming to reduce dimensions and find any important principal components. Then we use **Linear Regressions (LR)** to evaluate these principal components, proving that the data is effective for our model. Then we are able to add the transformed variables under the principal components axis into our Spreading Model for modification.

Eventually, according to the results above, we figure out a comprehensive and feasible **Strategy Program** called **"SSD"**. The whole strategy program can be divided into three parts: **Supply Control, Spread Control, and Demand Control**. After being tested by our promoted model, these strategies are proven to be significantly effective in controlling and preventing the opioid crisis.

Keywords: SOS; Back-Propagation Neutral Network (BPNN); Outlier Detection; Principal Component Analysis (PCA); Deep Learning Approach Auto-Encoder; Linear Regression (LR); Strategy Program "SSD"

SOS: Spreading Model, Optimization, and Strategy Program for the Opioid Crisis

January 29, 2019

Summary

As the deteriorating Opioid Crisis overwhelms the United States, the DEA/National Forensic Laboratory Information System (NFLIS) are striving to control the nationwide spread of opioid overdose indidents. In this paper, we formuate a concrete and novel framework, named **SOS (Spreading Model, Optimization, and Strategy Program for the Opioid Crisis)**, seeking for new strategies to combat the exacerbating opioid crisis.

Based on the report provided, we create a Spreading Model of opioid incidents, focusing on the individual counties located in five U.S. states: Ohio (OH), Kentucky (KY), West Virginia (WV), Virginia (VA), and Pennsylvania (PA). We first get the longitude and latitude of every county and utilize a **Back-Propagation Neutral Network (BPNN)** to characterize the reported opioid and heroin incidents in and between the five states and their counties from 2010 to 2017. Based on this initial model, we next adopt **Outlier Detection** to identify some most possible locations where specific opioid accidents might have started, and the trending opioids crisis in the future of the five states.

Furthermore, we combine **Principal Component Analysis (PCA)** and **Deep Learning Approach Auto-encoder (AE)** techniques to analyze the U.S. Census Socio-economic data provided, aiming to reduce dimensions and find any important principal components. Then we use **Linear Regressions (LR)** to evaluate these principal components, proving that the data is effective for our model. Then we are able to add the transformed variables under the principal components axis into our Spreading Model for modification.

Eventually, according to the results above, we figure out a comprehensive and feasible **Strategy Program** called **"SSD"**. The whole strategy program can be divided into three parts: **Supply Control, Spread Control, and Demand Control**. After being tested by our promoted model, these strategies are proven to be significantly effective in controlling and preventing the opioid crisis.

Keywords: SOS; Back-Propagation Neutral Network (BPNN); Outlier Detection; Principal Component Analysis (PCA); Deep Learning Approach Auto-Encoder; Linear Regression (LR); Strategy Program "SSD"

Contents

1 Introduction

1.1 Problem Background

The Opioid Crisis means the rapid increase in the use of prescription and non-prescription opioid drugs in the United States beginning in the late 1990s and continuing throughout the past two decades [1]. Nowadays, it has developed into a nationwide crisis sweeping across the United States. For instance, using the data supplied with this problem, we analyze just heroin identification counts during years 2010-2017 in Figure 1, which to some extent demonstrate the rampant drug problem in these five states, especially in Ohio (OH) and Pennsylvania (PA).See appendix A.1 for the complete code.

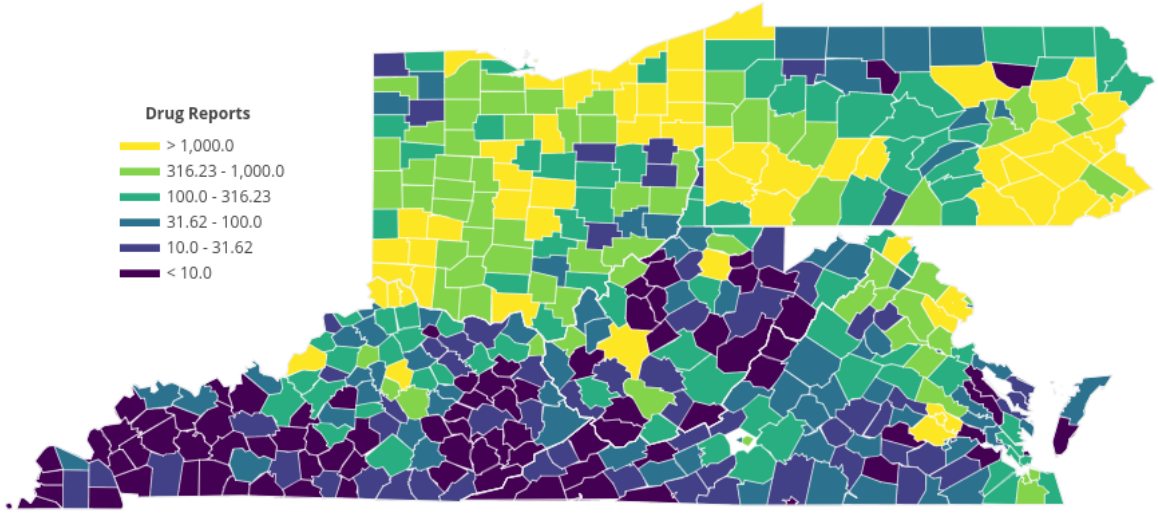


Figure 1: Heroin identification counts in years 2010-2017 in each of the counties from five states: Ohio (OH), Kentucky (KY), West Virginia (WV), Virginia (VA), and Pennsylvania (PA).

Studies show that the increase in opioid overdose deaths has been dramatic, and opioids are now responsible for 49,000 of the 72,000 drug overdose deaths overall in the US in 2017 [2]. The rate of prolonged opioid use is also increasing globally, threatening not only Americans' health but also the U.S. economy in many aspects. Consequently, president Donald Trump declared the country's opioid crisis a "national emergency" [3].

Currently, the U.S. government has payed great attention and taken a bunch of measures on this issue [4]. While the U.S. Centers for Disease Control (CDC) continue to fight the opioid overdose epidemic, simply enforcing existing laws is still a complex challenge for the Federal Bureau of Investigation (FBI), and the U.S. Drug Enforcement Administration (DEA), among others [5]. Therefore, they need investigate the spread and characteristics of the opioids and heroin incidents in the United States, so that they can develop their strategies to better control and prevent the deteriorating opioids overdose situation.

1.2 Our Work

In this paper, we propose a novel framework, named SOS (Spreading Model, Optimization, and Strategy Pragram for the Opioid Crisis), seeking for new strategies to combat the exacerbating opioid crisis. The framework of SOS is shown in Figure 2.

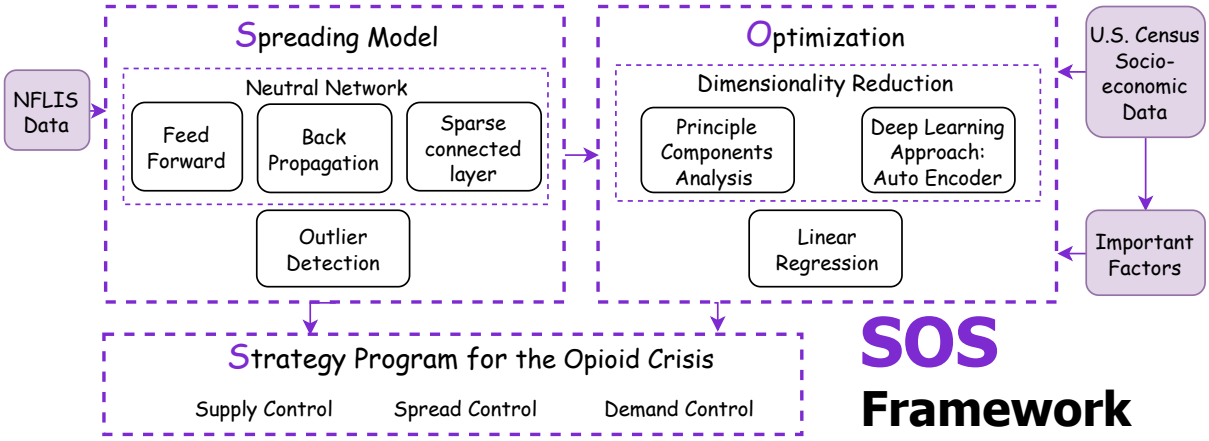


Figure 2: Framework of our model SOS: Spreading Model, Optimization, and Strategy Program for the Opioid Crisis.

We can divide our SOS framework in Figure 2 into following steps:

- **Spreading Model**: Based on the report provided, we focus on the individual counties located in five U.S. states: Ohio (OH), Kentucky (KY), West Virginia (WV), Virginia (VA), and Pennsylvania (PA). We first get the longitude and latitude of every county and build a **Back-Propagation Neutral Network (BPNN) Model [6]** to describe the spread and characteristics of the reported opioid incidents over time. With this model, we identify some possible locations where specific opioid use might have started in each of the five states.
- **Optimization**: Furthermore, we analyze the U.S. Census Socio-Economic data provided with two techniques: **Principal Component Analysis (PCA)** and **Learning Approach: Auto Encoder [6]** , aiming to reduce the dimensionalities and find any important principal components. Then we make **Linear Regressions** to test the validity of these components, with which we add to our Spreading Model.
- **Strategy Program for the Opioid Crisis**: Eventually, we identify a comprehensive and feasible Strategy Program for countering the opioid crisis. It contains three aspects: **Supply Control, Spread Control, and Demand Control(SSD)** After being tested by our model, these strategies are proven to be significantly effective in controlling the opioid crisis.

2 Assumptions

First and foremost, we make some basic assumptions and explain their rationales.

Assumption 1. *Each state pays great attention to the drug cases and establishes common goals to overcome the opioid crisis.*

This assumption is the premise of our work, because our goals as well as actions make sense only if every state strives to attack the drug problem.

Assumption 2. *The spread of the opioid and heroin incidents only takes place between two counties that are close to each other.*

To simplify this problem, we only take short-distance spread between counties into consideration, and omit long-distance smuggling of opioid.

Assumption 3. *There will be no sudden enactment of strict laws or regulations on drug control.*

The sudden changes of laws on drug controls will change the amount of the the cases during the period of time. There is no sign for changing of the laws, so that the assumption is reasonable. In that case, our data can be treated stable.

Assumption 4. *The drug identification data and the county location data are reliable to a certain extent.*

Although the data can not be as complete as the fact and some statistical errors are inevitable, we make this assumption to reach one valid solution.

3 Nomenclature

In this paper we use the nomenclature in Table 1 to describe our model. Other symbols that are used only once will be described later.

Table 1: Nomenclature.

Symbol	Definition
A_x	The longitude of point A
A_y	The latitude of point A
D_{ij}	The distance between any two state counties
R^2	The coefficient of determination
h_w	A vector function depends on the input-layer weights
y	Target result of the function h_w
Err_k	The kth component of the error vector $y - h_w$
Δ_k	A modified error $Err_k \times g'(in_k)$
w_k	A set of p-dimensional vectors of weights or loadings
$\mathbf{t}_{(i)}$	A new m-dimensional vector of principal component scores
$\mathbf{X^T X}$	A positive semidefinite matrix
$\mathbf{W}$	A p-by-p matrix whose columns are the eigenvectors of $\mathbf{X^T X}$
MLP	The Multilayer Perception
AE	Auto Encoder
PCA	Principal Component Analysis
LR	Linear Regression
BPNN	Back-Propagation Neutral Network

4 SOS: Spreading Model, Optimization, and Strategy Program for the Opioid Crisis

In this section, we will discuss all details about our model **SOS**. Generally, this model consists of three parts: Spreading Model for Opioid Incidents, Optimization with Socio-Economic Components, and Strategy Program for the Opioid Crisis.

4.1 Data Pre-Processing and Model Evaluation Criteria

4.1.1 Geometric Location and Distances between the Counties

To begin with, we consider the relative distance between any two counties as one of the factors that affect the spreading of Opioid Crisis. However, the data of geometry information of the counties listed is not given in the dataset. Thus, we write a short python program to get the geographical locations of all the counties in the above five states from the **Microsoft Bing Map API** [7], which is a tool used to search through OpenStreetMap data by name and address. The official coordinates of these 462 counties that we get are partly present as follows. See appendix A.2 for the detailed code.

Table 2: Part of the official coordinates of these 462 counties we get through Python from Microsoft Bing Map API.

State	County	latitude	longitude
KY	ADAIR	37.97293091	-86.84214783
KY	ALLEN	37.61523056	-82.72395325
KY	ANDERSON	37.01742172	-86.79804993
KY	BALLARD	38.23900986	-85.7456665
KY	BARREN	37.18073654	-86.62342072
KY	BATH	37.24637985	-82.90135956
KY	BELL	36.73059845	-83.67401123
KY	BOONE	37.51216888	-84.32032776
KY	BOURBON	37.05149841	-84.62284088

In that case, we can calculate the distance between each two counties. The earth is a near-standard ellipsoid with an equatorial radius of 6,378,140 km and a polar radius of 6,356,755 km, with an average radius of 6,371,004 km. If we assume that the earth is a perfect sphere, then its radius is the average radius of the earth, called R. If the zero degree longitude line is the basis, the surface distance between any two points on the earth's surface can be calculated based on the longitude and latitude of the two points. Let the longitude and latitude of point A be (A_x, A_y), and the longitude and latitude of point B be (B_x, B_y). Then, according to the Trigonometric Derivation, the equation 1and 2 for calculating the distance (D) between two points can be obtained.

$$C = \sin(A_y)\sin(B_y)\cos(A_x - B_x) + \cos(A_y)\cos(B_y), \tag{1}$$

$$D = R\arccos(C)\pi/180, \tag{2}$$

where C is a temporary variable and D is distance. Using the location data we got before, we can make a matrix that represents all the distance between any two counties in these five states.

4.1.2 Count of Drug Report by State

Besides, we rank the number of opioid incidents of each drug in each state county, and then we get the Table 3. Since there is no statistic significance for small amounts of data, we screen out significant data with statistically large amounts for the following modeling. See appendix A.4 for the complete code.

Table 3: Part of ranking Table according to the number of specific opioid incidents in each state county, sorted in descending order.

State	Substance Name	Drug Reports
KY	Hydrocodone	861
KY	Oxycodone	838
VA	Oxycodone	816
VA	Hydrocodone	746
VA	Heroin	716
OH	Heroin	682
OH	Oxycodone	682
OH	Hydrocodone	659
KY	Buprenorphine	642

4.1.3 Model Evaluation Criteria: Coefficient of Determination

In our model, we take Coefficient of Determination as our Evaluation Criteria.In statistics, the coefficient of determination, denoted R^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s) [8].

A data set has n values marked y_1,...,y_n (collectively known as y_i or as a vector $y = [y_1, ..., y_n]^T$), each associated with a predicted (or modeled) value f_1,...,f_n (known as f_i, or sometimes $\hat{y}_i$, as a vector f).

Define the residuals as $e_i = y_i$ - f_i (forming a vector $\vec{e}$). $\bar{y}$ is the mean of the observed data: $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, then the variability of the data set can be measured using three sums of squares formulas:

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \ , \ SS_{reg} = \sum_i (f_i - \bar{y})^2 \ , \ SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2.$$

Here we use the most general definition of the coefficient of determination as equation 3.

$$R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}} \tag{3}$$

The better the linear regression fits the data in comparison to the simple average, the closer the value of R^2 is to 1. A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.

4.2 Spreading Model for Opioid Incidents

4.2.1 Introduction

In order to describe how the opioids identification counts spread in and between the five states, we build a Back-Propagation Neutral Network model, using distance

matrix and last year's data as input. Back-Propagation is a supervised learning algorithm, for training Artificial Neural Networks, especially, multi-layer networks [9]. The weights that the minimum error occurs is then considered to be a solution to the learning problem.

However, the origin **BPNN** model is full-connected, which means, it doesn't take the distance between counties into consideration. The connections between two counties far away from each other can have a great impact on the learning time and effect of the model. Thus, we use a sparse-connected hidden layer between the input layer and output layer to solve this problem. Whether two neutrons are connected depends on whether the distance of the two counties they represent is under a certain threshold. Two neutrons of same counties are connected since their distance is zero.

In addition, the drugs reports can not only depend on the data of last year, but the data several years before as well. In practice, we found that using data of the previous two years could result in the better performance of the model. So we use a linear combination of them can try to learn two parameters for both years.

Here is an example structure of our model. Suppose there are five counties, named A, B, C, D, E, connected as shown in Figure 3.

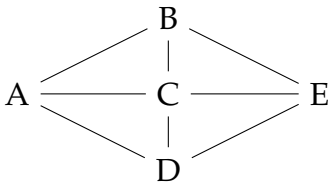


Figure 3: Diagrammatic drawing of five counties in our model

Then we can build a neutral network as shown in Figure 4. See appendix A.5 for the complete python code.

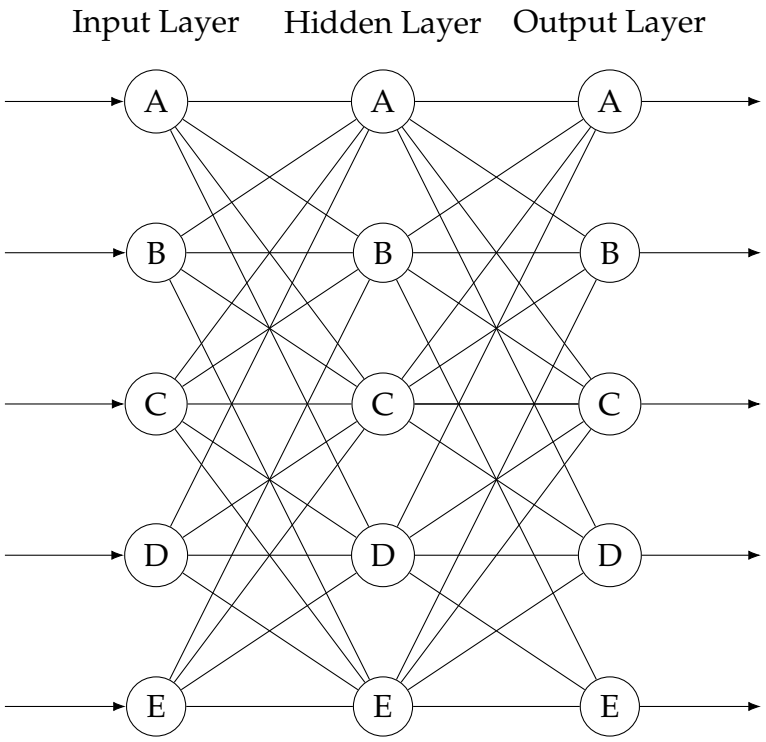


Figure 4: Diagrammatic drawing of BPNN model

4.2.2 Back-Propagation Neutral Network (BPNN) Model

Let h_w be a vector function and y be its target result. Then Err_k be the kth component of the error vector $y - h_w$.

Whereas a percentage network decomposes into m seperate learning problems for an m-output problem, this decomposition fails in multi-layer network. The vector h_w that returns depends on the all of the input-layer weights, so updates to those weights will depend on errors in the vector. Fortunately, this dependency is very simple in the case of any loss function that is additive across the components of the error vector $y - h_w$. For the L_2 loss, we have, for any weight w, equation 4.

$$\frac{\partial}{\partial w} Loss\left(w\right) = \frac{\partial}{\partial w} \left|y - h_w\left(x\right)\right|^2 = \frac{\partial}{\partial w} \sum_k \left(y_k - a_k\right)^2 = \sum_k \frac{\partial}{\partial w} \left(y_k - a_k\right)^2 \tag{4}$$

The major complication comes from the addition of hidden layers to the network. Whereas the error $y - h_w$ at the output layer is clear, the error at the hidden layers seems mysterious because the training data do not say what value the hidden nodes should have. Fortunately, it turns out that we can back-propagate the error from the output layer to the hidden layers. The back-propagation progress emerges directly from a derivation of the overall error gradient.

We can easily define a modified error $\Delta_k = Err_k \times g'\left(in_k\right)$, where in_k means the kth component of input, so that the weight update rule becomes the equation 5.

$$w_{j,k} \leftarrow w_{j,k} + \alpha \times a_j \times \Delta_k \tag{5}$$

To update the connections between the input units and the hidden units, we need to define a quantity analogous to the error term for output nodes. Here is where we do the error back-propagation. The idea is that hidden node j is "responsible" for some fraction of the error Δ_k in each of the output nodes to which it connects. Thus, the Δ_k values are divided according to the strength of the connections between the hidden node and the output node and are propagated back to provide the Δ_j values for the hidden layer. The propagation rule for the Δ values is showed in equation 6.

$$\Delta_j = g'\left(in_k\right) \sum_k w_{j,k} \Delta_k \tag{6}$$

Now the weight-update rule for the weights between the inputs and the hidden layer is essentially identical to the update rule for the output layer:

$$w_{j,k} \leftarrow w_{j,k} + \alpha \times a_j \times \Delta_k \tag{7}$$

4.2.3 Prediction and Outlier Detection

Outlier is a value that lies in a data series on its extremes, which is either very small or large and thus can affect the overall observation made from the data series. Outliers are also termed as extremes because they lie on the either end of a data series. We apply Outlier to look for the prediction of the the specific concerns and the source of the each drug crisis.

Let n be the number of data values in the data set. The Lower Quartile $Q1$ is the median of the lower half of the data set. The Upper Quartile $Q3$ is the median of the

upper half of the data set. The Interquartile range IQR is the spread of the middle 50% of the data values.

$$\begin{aligned} IQR &= Q3 - Q1 \\ Lower Limit &= Q1 - 1.5IQR \\ Upper Limit &= Q3 + 1.5IQR \end{aligned} \tag{8}$$

Therefore, any value that will be more than the upper limit or lesser than the lower limit will be the outliers. We identify that these points are trending to have an opioid crisis.

4.2.4 Source Deduction and Future Prediction: Spreading Model between Counties

As we mentioned before in 4.1.2 Table 3, Hydrocodone in the state KY and Buprenorphine in OH are the most often occurred cases by statistics. So we choose it as a sample for us to derive its source. Considering that we have just 8-year data, so we only predict 2 years forward and backward to ensure our results accurate. Thus, With data from 2010 to 2017 inputted into our model, we predict how the identification counts for the Hydrocodone cases distribute in KY in 2008, which is shown in Table 4a and Figure 5a. Similarly, we can predict Buprenorphine cases in Ohio (OH) in 2019, which is shown in Table 4b and Figure 5b.

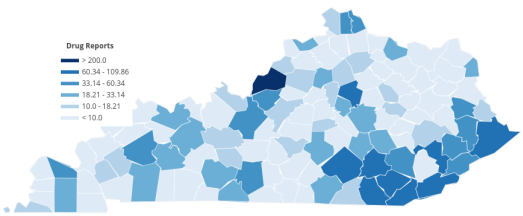
Table 4: Source Dedection and Future Prediction

(a) Source Dedection: Hydrocodone identification counts in 2008 in Kentucky (KY).

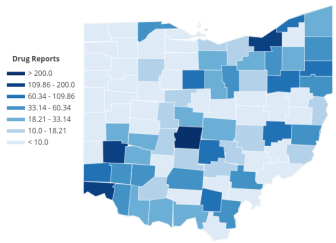
County	Drug Reports
JEFFERSON	1255.26
LAUREL	118.82
PERRY	112.61
BELL	87.37
HARLAN	88.33
PULASKI	85.35
FAYETTE	115.80
WHITLEY	64.73
PIKE	91.37
FLOYD	70.30
KNOX	61.78

(b) Future Prediction: Buprenorphine identification counts in 2019 in Ohio (OH).

County	Drug Reports
FRANKLIN	240.30
HAMILTON	155.90
MONTGOMERY	140.91
CUYAHOGA	139.50
FAIRFIELD	98.06
CRAWFORD	89.35
STARK	82.62
GUERNSEY	81.52
LAKE	78.01
COLUMBIANA	76.87
BUTLER	73.77



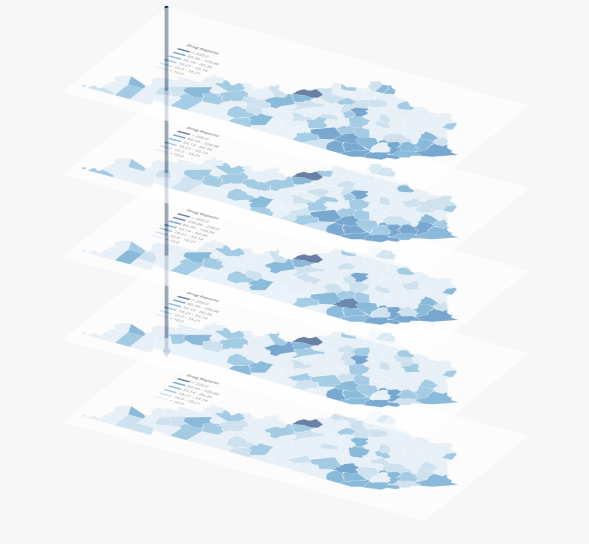
(a) Hydrocodone identification counts in 2008 in Kentucky (KY)



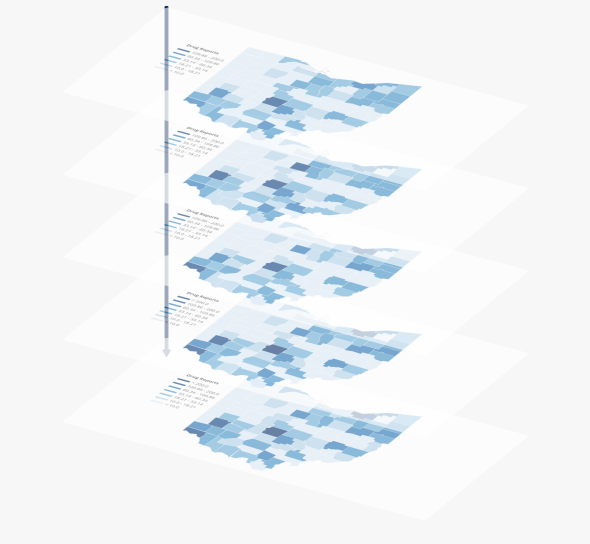
(b) Buprenorphine identification counts in 2019 in Ohio (OH)

Figure 5: Source Dedection and Future Prediction

Then we overlay such images generated from 2008 to 2012 in chronological order to show the spreading characteristics of Hydrocodone identification counts in Kentucky as Figure 6a shows. With this method, we identify some possible locations where specific opioid use might have started in each of the five states. Similarly, we can demonstrate the spreading characteristics of Buprenorphine identification counts in Ohio as Figure 6b shows.



(a) Hydrocodone identification counts spreading diagram from 2008 to 2012 in Kentucky



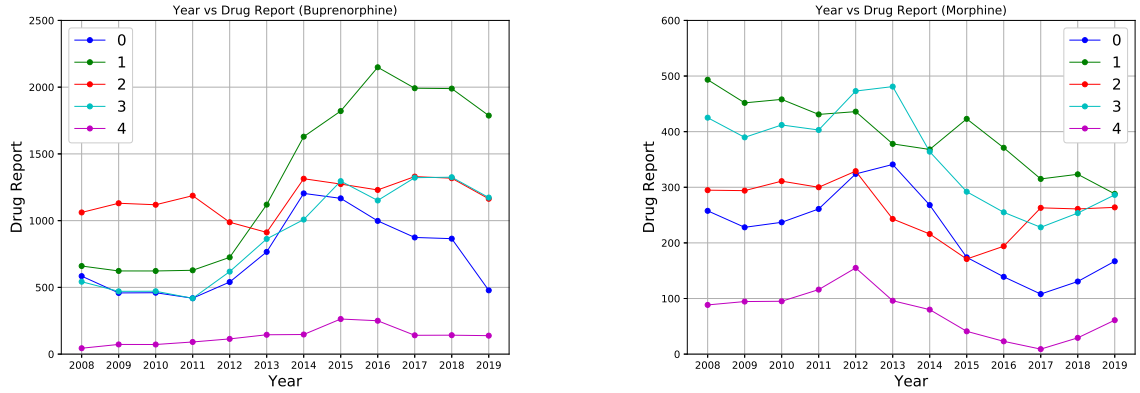
(b) Buprenorphine identification counts spreading diagram from 2014 to 2019 in Ohio

Figure 6: Source Dedection and Future Prediction model

4.2.5 Source Deduction and Future Prediction: Spreading Model of States

The Spreading Model of states can be easily modified to obtain source deduction and future prediction for each state, since the five states in the dataset are next to each other, we can consider they are full-connected in the neutral network. After removing the distance module in the previous model, we build a Spreading Model of states. See appendix A.5 for the complete python code (same file as the previous model).

For example, we take Buprenorphine and Morphine and predict the drug report on 2008-2009 and 2018-2019, the results are shown in Figure 7.



(a) Buprenorphine (b) Morphine

Figure 7: Buprenorphine and Morphine identification counts of five states in each year from 2008 to 2019.

4.3 Optimization: Modification using Socio-economic Component

4.3.1 Dimensionality Reduction

Technique 1: Principal Component Analysis (PCA)

Now, our model can basically describe the spread and characteristics of the opioid and identify some possible locations where specific opioid use might have started. Next, we consider whether any important factors from the U.S. Census socio-economic data provided can further modify our model so that it can explain how opioid use got to its current level, what contributes to the growth in opioid addiction, and analyze who is using/abusing it despite its known dangers.

When we process the U.S. Census socio-economic data, we first adopt the Principal component analysis (PCA), which is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. If there are n observations with p variables, then the number of distinct principal components is $\min(n-1, p)$.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a data matrix, $\mathbf{X}$, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of feature (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of p-dimensional vectors of weights or loadings $\mathbf{w}_{(k)} = (w_1, \ldots, w_p)_{(k)}$ that map each row vector $\mathbf{x}_{(i)}$ of X to a new vector of principal component scores $\mathbf{t}_{(i)} = (t_1, \ldots, t_m)_{(i)}$, given by

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)} \qquad \text{for} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, m$$

in such a way that the individual variables $t_1, \ldots, t_m$ of t considered over the data set successively inherit the maximum possible variance from $\mathbf{x}$, with each loading vector

w constrained to be a unit vector.

In order to maximize variance, the first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2 \right\} \tag{9}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \{ \|\mathbf{X}\mathbf{w}\|^2 \} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \right\} \tag{10}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \arg\max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\} \tag{11}$$

The quantity to be maximised can be recognised as a Rayleigh quotient. A standard result for a positive semidefinite matrix such as $\mathbf{X^T X}$ is that the quotient's maximum possible value is the largest eigenvalue of the matrix, which occurs when w is the corresponding eigenvector.

With $\mathbf{w}_{(1)}$ found, the first principal component of a data vector $\mathbf{x}_{(1)}$ can then be given as a score $t_{1(1)} = \mathbf{x}_{(1)} \cdot \mathbf{w}_{(1)}$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{\mathbf{x}_{(1)} \cdot \mathbf{w}_{(1)}\}\mathbf{w}_{(1)}$.

The kth component can be found by subtracting the first k_1 principal components from $\mathbf{X}$:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{w}_{(s)}\mathbf{w}_{(s)}^{\mathrm{T}} \tag{12}$$

and then finding the loading vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \|\hat{\mathbf{X}}_k \mathbf{w}\|^2 \right\} = \arg\max \left\{ \frac{\mathbf{w}^T \hat{\mathbf{X}}_k^T \hat{\mathbf{X}}_k \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\} \tag{13}$$

It turns out that this gives the remaining eigenvectors of $\mathbf{X^T X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues. Thus the loading vectors are eigenvectors of $\mathbf{X^T X}$.

The kth principal component of a data vector $\mathbf{x}_{(i)}$ can therefore be given as a score $t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$ in the transformed co-ordinates, or as the corresponding vector in the space of the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}\}\mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$ is the kth eigenvector of $\mathbf{X^T X}$.

The full principal components decomposition of $\mathbf{X}$ can therefore be given as

$$\mathbf{T} = \mathbf{X}\mathbf{W} \tag{14}$$

where $\mathbf{W}$ is a p-by-p matrix whose columns are the eigenvectors of $\mathbf{X^T X}$. The transpose of $\mathbf{W}$ is sometimes called the whitening or sphering transformation.

See appendix A.11 for the complete python code.

Technique 2: Deep Learning Approach: Auto Encoder

An autoencoder is an artificial neural network used for unsupervised learning of efficient codings. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data. Some of the most powerful AI in the 2010s involves stacking sparse autoencoders in a deep learning network.

Architecturally, the simplest form of an autoencoder is a feedforward, non-recurrent neural network very similar to the multilayer perception (MLP), having an input layer, an output layer and one or more hidden layers connecting them, but with the output layer having the same number of nodes as the input layer, and with the purpose of reconstructing its own inputs (instead of predicting the target value Y given inputs X. Therefore, autoencoders are unsupervised learning models.

An autoencoder always consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and ψ such that:

$$\begin{gathered}\phi : \mathcal{X} \to \mathcal{F} \\ \psi : \mathcal{F} \to \mathcal{X} \\ \phi, \psi = \underset{\phi,\psi}{\arg\min} \|X - (\psi \circ \phi)X\|^2 \end{gathered} \tag{15}$$

In the simplest case, where there is one hidden layer, the encoder stage of an autoencoder takes the input $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ and maps it to $\mathbf{z} \in \mathbb{R}^p = \mathcal{F}$:

$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{16}$$

This image $\mathbf{z}$ is usually referred to as code, latent variables, or latent representation. Here, σ is an element-wise activation function such as a sigmoid function or a rectified linear unit. $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector. After that, the decoder stage of the autoencoder maps $\mathbf{z}$ to the reconstruction $\mathbf{x}'$ of the same shape as $\mathbf{x}$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}') \tag{17}$$

where $\sigma', \mathbf{W}',$ and $\mathbf{b}'$ for the decoder may differ in general from the corresponding $\sigma, \mathbf{W},$ and $\mathbf{b}$ for the encoder, depending on the design of the autoencoder.

Autoencoders are also trained to minimise reconstruction errors (such as squared errors):

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2 \tag{18}$$

where $\mathbf{x}$ is usually averaged over some input training set.

If the feature space $\mathcal{F}$ has lower dimensionality than the input space $\mathcal{X}$, then the feature vector $\phi(x)$ can be regarded as a compressed representation of the input x. If the hidden layers are larger than the input layer, an autoencoder can potentially learn the identity function and become useless. However, experimental results have shown that autoencoders might still learn useful features in these cases.

4.3.2 Linear Regression (LR) for Principle Component

After we choose the number of components to express the use or trends-in-use according to the U.S. Census socio-economic data, we apply linear regression to test if

the components we choose can give a satisfying result that is close to the reports given by the NFLIS data. In this way, we can a score of this linear regression. The closer it approaches 1, the better the components show the fact.

Given a data set $\{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable , an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form as the following shows.

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = x_i^T \beta + \varepsilon_i \tag{19}$$

The superscript T denotes the transpose, so that $x_i^T \beta$ is the inner product between vectrs x_i and β.

Often these n equations are stacked together and written in matrix notation as the following.

$$y = X\beta + \varepsilon \tag{20}$$

- y is a vector of observed values $y_i (i = 1, \cdots, n)$ of the variable called the regressand, endogenous variable, response variable, measured variable, criterion variable, or dependent variable. This variable is also sometimes known as the predicted variable. X may be seen as a matrix of row-vectors
- x_i or of n-dimensional column-vectors X_j, which are known as regressors, exogenous variables, explanatory variables, covariates, input variables, predictor variables, or independent variables (not to be confused with the concept of independent random variables).
- β is a $(p+1)$-dimensional parameter vector, where β_0 is the intercept term (if one is included in the modelotherwise β is p-dimensional). Its elements are known as effects or regression coefficients (although the latter term is sometimes reserved for the estimated effects).
- ε is a vector of values ε_i. This part of the model is called the error term, disturbance term, or sometimes noise (in contrast with the "signal" provided by the rest of the model).

4.3.3 Results for Dimension Reduction and Linear Regression

We have selected the most important factors from the 596 dimensions that the U.S. Census socio-economic data that provides by dimensional reduction. And then, we use linear regression to test our selection and get a score for each selection. In this way, we can get a satisfying set of components.

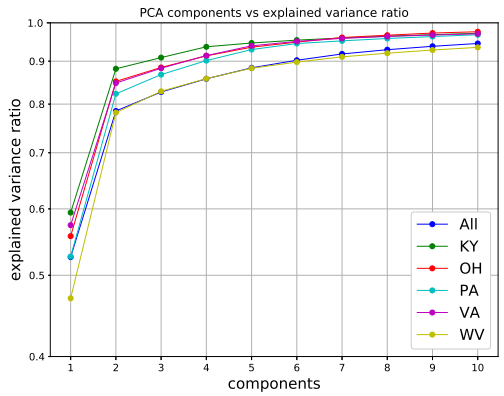
According to PCA, we can get the variance ratio, which shows the ratio of the variance value of each principal component to the total variance value after dimension reduction. The larger the value of the variance ratio, the more important the principal component. Actually, the more components we take into account, the larger variance ratio that we can get. However, what we pursue is not only the completeness characterization of the fact ,but also the simplification of the vast dimension. So we need to

get an appropriate set of components that can characterize all the information as well as simplify this problem.

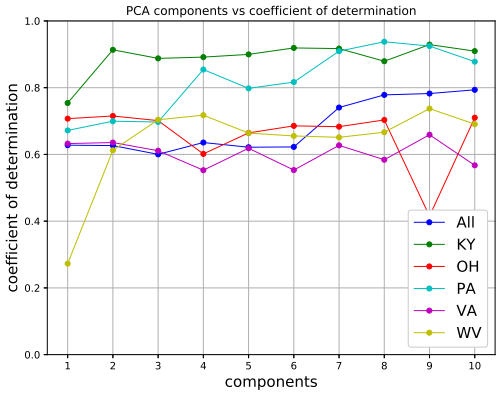
According to LR, we can get a score for the fitting. Actually, the score is the R^2 of linear regression results and the fact data provided by the NFLIS Data, which shows how closer the results we get from the components reach the fact. The closer it approaches 1, the better our fitting according to the chosen components.

In addition, to get a good result, we also need to select the way to deal with the data given by the U.S. Census socio-economic data. So we apply the PCA as well as LR to the these data in four ways, and you can see appendix A.12 for the complete python code. Here we show the results by diagrams:

(1) Analyze all the data over the years ranging from 2010 to 2016 of the five states respectively, and the results are shown in Figure 8.



(a) The ratio of data retained after dimensional reduction among five states in 2010-2016 against different components



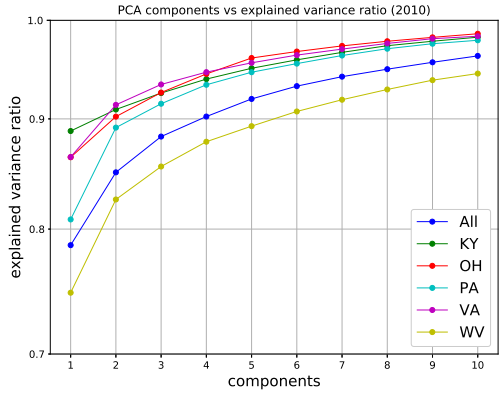
(b) The R^2 of linear regression results among five states in 2010-2016 against different components

Figure 8: PCA and LR results of data among five states in 2010-2016 against different components

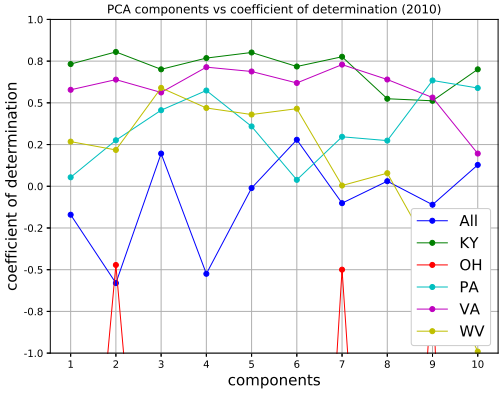
From (a) in Figure 8, we can see the curve for a single state raises as the number of the components increases. When we choose four components to describe this problem, the superimposed explained variance ratio is almost close to 1. That is to say that under this circumstance, the results that we get from the fitting is very close to the fact.

From (b) in Figure 8, we can see the curves for the five years are almost stable as the number of the components increase, except some special cases. As for one component for the WV state, the low coefficient of determination ascribes that there is only one component to characterize the problem. As for nine components for the OH state, the low coefficient of determination may attribute to over-fitting.

(2) Analyze the data of 2010 of the five states respectively, and the results are shown in Figure 9.



(a) The ratio of data retained after dimensional reduction among five states in 2010 against different components

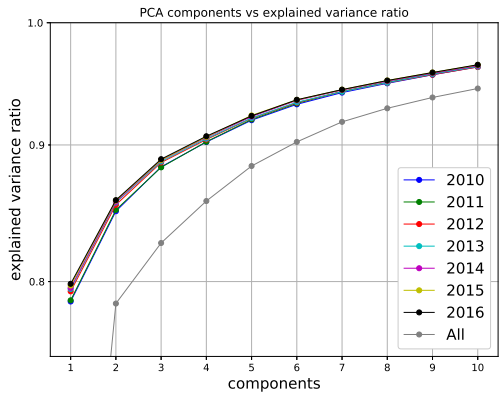


(b) The R^2 of Linear Regression results among five states in 2010 against different components

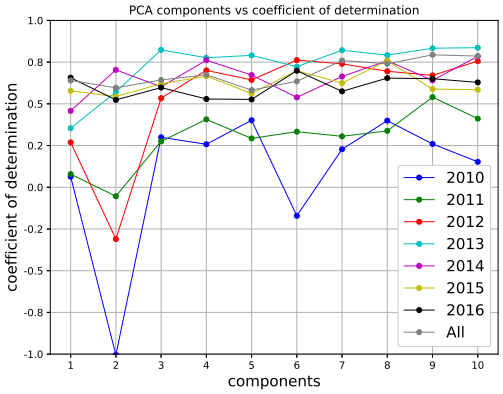
Figure 9: PCA and LR results of data among five states in 2010 against different components

From (a) in Figure 9, we can see the ratio shows the similar characteristics to (a) in Figure 8, which means an excellent characterization of this problem with only four components. But when it comes to the evaluation part, from (b) in Figure 9, we can directly see that the fitting is not as stable as the last one.

(3) Analyze the data of all the five states of 2010 to 2016 respectively, and the results are shown in Figure 10.



(a) The ratio of data remained after dimensional reduction among five states in 2010-2016

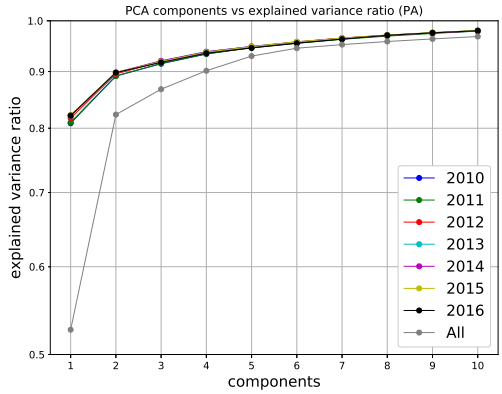


(b) The R^2 of Linear Regression results of data in 2010-2016 against different aggregations

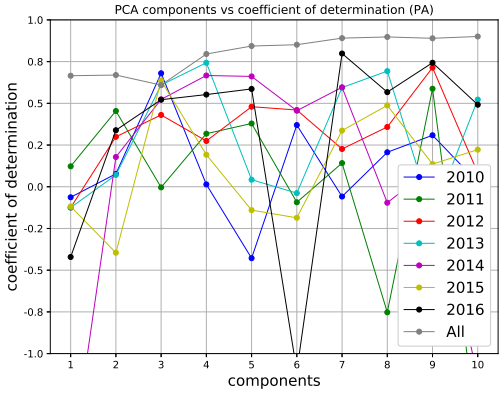
Figure 10: PCA and LR results of data in 2010-2016 against different aggregations

From (a) in Figure 10, we can see the curve for a single year raises as the number of the components increases almost in the same way. However, the superimposed explained variance ratio gets close to 1 when the number of the components raises to 6 or more, which indicates that it does not do well in simplification. Moreover, the coefficient of determination that is shown in (b) of Figure 10 is unstable.

(4) Analyze the data of the state PA of 2010 to 2016 respectively, and the results are shown in Figure 11.



(a) The ratio of data after dimensional reduction in PA in 2010-2016



(b) The R^2 of Linear Regression results in PA over years against different aggregations

Figure 11: PCA and LR results of data in PA in 2010-2016 against different aggregations

From (a) in Figure 11, the curve shows similar characteristics to the above ones and also does well in characterize through only few components. But (b) of Figure 11 also shows its instability.

From the visualization and analysis that we have shown above, it can be concluded that when we choose all the data over the years ranging from 2010 to 2016 of the five states respectively, we can gain a great, simple, and stable dimensional reduction with only 4 components.

For example, we select the U.S. Census socio-economic data of KY state ranging from 2010 to 2016. In PCA, the variance ratio reaches a threshold of about 0.01 after the fourth dimension. The first four variance ratios are

$$0.59396827, 0.28748557, 0.02754973, 0.02732665.$$

From these four dimension, we can get the score of about 0.92 by linear regression, which shows a good result of the dimension reduction.

4.3.4 Results for Optimizied SOS Model

Based on the results of Dimension Reduction and Linear Regression in section 4.3.3, we add these four principal components into our BPNN model for modification. With this Optimizied SOS Model, we do as section 4.2.4 do and get the Table 5a and Table 5b

Table 5: Results Table for Optimizied SOS Model

(a) Hydrocodone identification counts in 2008 in Kentucky (KY) using modified SOS model

County	Drug Reports
JEFFERSON	1158.60
LAUREL	103.17
PERRY	98.59
BELL	86.35
HARLAN	85.24
PULASKI	78.45
FAYETTE	77.75
WHITLEY	68.65
PIKE	68.62
FLOYD	61.29

(b) Buprenorphine identification counts in 2019 in Ohio (OH) using modified SOS model

County	Drug Reports
FRANKLIN	240.30
HAMILTON	155.90
MONTGOMERY	140.92
CUYAHOGA	139.51
FAIRFIELD	98.06
CRAWFORD	89.35
STARK	82.62
GUERNSEY	81.52
LAKE	78.01
COLUMBIANA	75.36

Finally, we apply the model to all of the five states so that we can analysis the coefficient of determination between the origin model and the new model with PCA optimization. The results are shown in Table 6

State	Substance	Origin Model	Optimized Model
KY	Hydrocodone	0.8957447139498953	0.9120307701827662
OH	Hydrocodone	0.8501898003702214	0.8641988375370137
PA	Hydrocodone	0.9319499913029298	0.9325942856975605
VA	Hydrocodone	0.8191016628092015	0.8374162880363936
WV	Hydrocodone	0.6797258899970731	0.7051639551372615
KY	Buprenorphine	0.7697562356106809	0.7998899587972155
OH	Buprenorphine	0.8720902736185301	0.8807783772009824
PA	Buprenorphine	0.9379855180669447	0.9418626513609050
VA	Buprenorphine	0.8720028255610893	0.8832633164948145
WV	Buprenorphine	0.12985720067610063	0.14259618407280017

Table 6: The coefficient of determination of the origin model vs the new model with PCA optimization.

We can found that the optimization have a good effect on the value of R^2, however, on the last row, Buprenorphine in WV State, the model doesn't fit the data well. This is probably because of the lack of data (only 234 data, and most of them are relatively small) of this kind of opioid in this state.

4.4 Strategy Program for the Opioid Crisis

4.4.1 SSD (Supply Control, Spread Control, and Demand Control)

Understanding that certain levels of drug use are inevitable, so we should focus on minimizing adverse effects associated with drug use rather than stopping the behavior itself. In the context of the opioid epidemic, we propose the **Strategy Program "SSD" (Supply Control, Spread Control, Demand Control)** that are designed to improve health outcomes and reduce overdose deaths.

It can be divided into three aspects:

Supply Control

- Requiring manufacturers of long-acting opioids to sponsor educational programs for prescribers. Through these educational programs, we can control opioid supply and help deter off-label and over-prescribing, making a threshold to those who want to abuse the opioids [10].
- We need to strengthen cooperation with countries such as Colombia and Mexico, to crack down on cross-border drug crimes.

Spread Control

- Strengthen public health data reporting and collection to improve the timeliness and specificity of data and to inform a real-time public health response as the opioid epidemic evolvesso that our Spreading Model can forecast the opioid epidemic more precisely [11].
- Strengthening Public Health Data and Reporting. Timely, high-quality data help both public health officials and law enforcement understand the extent of the problem and how it is evolving, develop interventions, focus resources where they are needed most, and evaluate the success of prevention and response efforts.

Demand Control

- Advance the practice of pain management to enable access to high-quality, evidence-based pain care that reduces the burden of pain for individuals, families, and society while also reducing the inappropriate use of opioids and opioid-related harms [12].
- Only by making opioid control and management more rigorous and formal can we effectively combat illicit drug abuse.

4.4.2 Effectiveness of our Strategy Program

Supply Control

Corresponding to our SOS model, controlling Supply means setting a **threshold function** for input in our model. Once certain opioid reaches its threshold, it would be hindered strongly by FBI, which makes sense in reality. Therefore, the Corresponding output will decrease significantly.

Spread Control

Corresponding to our SOS model, controlling Spread means setting a **punishing function** in our model based on the distance between each two counties. The farther apart the two counties are, the greater the spreading costs they pay, which also makes sense in reality.Therefore, the Corresponding output will decrease significantly.

Demand Control

Corresponding to our SOS model, controlling Demand means setting a **resisting function** in our every neuron, since people are less convenient to have access to opioids. By this way, we can simulate actual situations, which prove to be effective in countering opioids crisis.Therefore, the Corresponding output will decrease significantly.

5 Model Analysis

5.1 Sensitivity Analysis

- In our Back-Propagation Neutral Network (BPNN) Model, there is no parameter, but a sparse-connected hidden layer between the input layer and output layer to solve this problem. Thus, the robustness of our model is only dependent on data scale.
- In our Principal Component Analysis (PCA) and Linear Regression progress, we did the sensitivity analysis in section 4.3.3.
- As for Strategy Program "SSD", we have no extra time to do sensitivity analysis, but we did a comprehensive theoretical analysis concerning every possible factors.

5.2 Strengths and Weaknesses

5.2.1 Strengths

- **Effective and Uniform Data Extraction:** We uniformly use the library pandas in python to evaluate and filter the vast data for each problem. In this way, we apply the most appropriate data to each part and get a better result effectively.
- **High Characterization and Generalizability:** We design the model by BPNN that fits not only all the data but also each state of this problem.
- **Innovative Modeling with BPNN:** We apply BPNN to construct our model with highly self-learning and highly self-adapting abilities, which is good for us to solve this problem.
- **Bilateral Prediction by BPNN:** Our model can predict not only when and where specific concerns may occur but also the source of the crisis.
- **Alternative Methods for Dimensionality Reduction:** We introduced two methods: PCA and AE for dimensionality reduction and compare their results.
- **Prudential Test of the Model:**After the dimensionality reduction, we apply the components that we get to LR and get a fitting result. Through comparison, we evaluate whether our dimensionality reduction is good.
- **Quantified and Rational Goals:** We set quantified goals strictly based on optimization theory.

5.2.2 Weaknesses

- **No Verification of Raw Data:** We have no guarantee of the accuracy of given data , and the data is not complete.
- **No Involvement of Other States:** We do not consider states other than OH, KY, WV, VA, and PA, due to the lack of relevant data.
- **No Accurate Geographical Information:** We use the the official coordinates of counties to represent its geographical location. And we have no guarantee of the accuracy of the data from the Search (Nominatim) API [7].

6 Conclusion

In this paper,we propose a novel framework called **SOS** (Spreading Model, Optimization, and Strategy Program for the Opioid Crisis). First we create a **Back Propagation Neutral Network (BPNN) Model** to describe the spread and characteristics of the opioid incidents in and between the five states over time. With this model we identified possible locations where specific opioid use might have emerged in each of the five states. Second we adopt two different techniques: **Principal Component Analysis (PCA)** and **Deep Learning Approach: Auto-Encoder**, to extract principal components from the U.S. Census Socio-Economic data. After testing the validity of these components with **Linear Regression**, we add to our Spreading Model for modification. Third we identify a **Strategy Program "SSD"** for countering the opioid crisis. Tested by our model, these strategies are proven to be significantly effective in overcoming the opioid crisis. Finally, we conduct sensitivity analysis of some parameters in our model and discuss the strengths and weakness of our work.

MEMORANDUM

To: Chief Administrator, DEA/NFLIS Database

From: Team # 1920446

Date: Jananry 29, 2019

Subject: The Opioids Model: Analysis for Spread, Idetification for Source, Prediction for Future, and Strategy For Control

Honorable Chief Administrator, DEA/NFLIS Database,

Currently, the opioid crisis has overwhelmed the whole America, urging us to lay great emphasis on controlling the spreading opioid incidents. As the person in charge, our team has made a comprehensive study on countering opioid crisis based on the reported synthetic opioid and heroin incidents (cases) from 2010 to 2017 in and between the five states: Ohio (OH), Kentucky (KY), West Virginia (WV), Virginia (VA), and Pennsylvania (PA). We further analyze the U.S. Census Socio-Economic data to modify the initial model. Here come our study results:

Spreading Model

Here we utilize a Back-Propagation Neutral Network (BPNN) to build this model, characterizing the basic evolving and spreading trends of the opioid and heroin incidents. Thus, we forecast the opioid accidents happened in the following two years, and find that if we do not take effective measures at once, most of opioids incidents will increase sharply, which demonstrates the urgence of solving the opioid abuse problem right now.

In order to identify any possible locations where specific opioid use might have started, we adopt Outlier Detection and identify the outliers as the trending locations.

Optimization

In this part, we combine Principal Component Analysis (PCA) and Learning Approach Auto-Encoder two techniques to analyze the U.S. Census Socio-economic data provided. We successfully select the most important 4 factors from the 596 dimensions in data. And then, we use Linear Regression to test our selection and get a score for each selection, Which shows the similarity to the fact. In this way, we get a satisfying set of components added to our model for modification.

Strategy Program

Eventually, according to the results above, we figure out a comprehensive and feasible Strategy Program called "SSD". The whole strategy program can be divided into three parts: Supply Control, Spread Control, and Demand Control. After being tested by our promoted model, these strategies are proven to be significantly effective in controlling and preventing the opioid crisis. Concrete Strategies are as follows:

- **Supply Control:** To require manufacturers of long-acting opioids to sponsor educational programs for prescribers, making a threshold to those who want to

abuse the opioids.To strengthen cooperation with countries such as Colombia and Mexico, to crack down on cross-border drug crimes.

- **Spread Control:** To strengthen public health data reporting and collection to improve the timeliness and specificity of data and to inform a real-time public health response as the opioid epidemic evolves.
- **Demand Control:** To advance the practice of pain management to enable access to high-quality, evidence-based pain care that reduces the burden of pain for individuals, families, and society while also reducing the inappropriate use of opioids and opioid-related harms. To make opioid control and management more rigorous and formal for effectively combating illicit drug abuse.

The above is the summary of our study. We sincerely hope that it will provide you with useful information.

Thanks!

References

[1] "Opioid_epidemic," Wikipedia. [Online]. Available: https://en.wikipediwikipedia/Opioid_epidemic

[2] "This is nida: Opioids," NIH. [Online]. Available: https://www.drugabuse.gov/related-topics/trends-statistics/overdose-death-rates

[3] "Trump says opioid crisis is a national emergency pledges more money and attention," August 10, 2017. [Online]. Available: https://www.washingtonpost.com/politics/trump-declares-opioid-crisis-is-a-national-emergency-pledges-more-money-and-attention/2017/08/10/5

[4] P. W. Corrigan and K. Nieweglowski, "Stigma and the public health agenda for the opioid crisis in america," *International Journal of Drug Policy*, vol. 59, pp. 44 – 49, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0955395918301774

[5] "Confronting opioids," Centers for disease control and prevention. [Online]. Available: https://www.cdc.gov/features/confronting-opioids/index.html

[6] F. V. Nurçin, E. Imanov, A. In, and D. U. Ozsahin, "Lie detection on pupil size by back propagation neural network," *Procedia Computer Science*, vol. 120, pp. 417 – 421, 2017, 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917324705

[7] "Bing maps," Microsoft. [Online]. Available: https://www.bingmapsportal.com/

[8] "Coefficient_of_determination," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination

[9] S. J.Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson India Education Services, 2017.

[10] D. Fuster and R. Muga, "The opioid crisis," *Medicina Clínica (English Edition)*, vol. 151, no. 12, pp. 487 – 488, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2387020618304480

[11] S. Awsare, C. Havens, and J. Lippi, "Facing the opioid crisis: practical, effective actions we can take," *Gastroenterology*, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016508519300502

[12] D. A. Seigerman, K. Lutsky, M. Kwok, S. Sodha, D. Fletcher, D. Mazur, and P. K. Beredjiklian, "Whats new in the battle against the opioid crisis in hand surgery: Aăreview," *Journal of Hand Surgery Global Online*, vol. 1, no. 1, pp. 28 – 31, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2589514118300458

Appendices

Appendix A Python Code

A.1 map.py

```
import plotly.figure_factory
import plotly

import pandas as pd
import numpy as np

df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
    sheet_name='Data')
fips_dict = dict()
for index, row in df.iterrows():
    fips_dict[row['FIPS_Combined']] = 0

df_2010_Heroin = df[(df['SubstanceName'] == 'Heroin')]

for index, row in df_2010_Heroin.iterrows():
    fips_dict[row['FIPS_Combined']] += row['DrugReports']

fips = list(fips_dict.keys())
values = list(fips_dict.values())

plotly.tools.set_credentials_file(username='tc-imba',
    api_key='xNu6lsfY6Twz6LfUmjHa')
endpts = list(np.geomspace(10, 1000, 5))

fig = plotly.figure_factory.create_choropleth(
    fips=fips, values=values,
    scope=['VA', 'OH', 'PA', 'KY', 'WV'],
    binning_endpoints=endpts,
    county_outline={'color': 'rgb(255,255,255)', 'width':
        0.5},
    legend_title='Drug Reports'
)

fig['layout']['legend'].update({'x': 0.25, 'y': 0.75})
fig['layout']['annotations'][0].update({'x': 0.12, 'y': 0.8,
    'xanchor': 'left'})

# plotly.offline.plot(fig, filename='map.html')
plotly.plotly.plot(fig, filename='test',
    fileopt='overwrite', auto_open=False)
# # plotly.plotly.image.save_as(fig, 'test.png', width=1920,
    height=1080)
```

A.2 geocode.py

```
from geopy.geocoders import Bing
from geopy.extra.rate_limiter import RateLimiter

import pandas as pd
from tqdm import tqdm

tqdm.pandas()
```

```

geolocator =
    Bing(api_key='AtlqFZkg8aISlN-CNOpURU3oLthMK6g166C9gDCO1sc9Cl5njVi1N
geocode = RateLimiter(geolocator.geocode,
    min_delay_seconds=0.1)

df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
    sheet_name='Data')

df = df.groupby('FIPS_Combined').first().reset_index()

df['name'] = df['COUNTY'] + ', ' + df['State']
df['location'] = df['name'].progress_apply(geocode)
df['latitude'] = df['location'].apply(lambda loc: loc and
    loc.latitude or 0)
df['longitude'] = df['location'].apply(lambda loc: loc and
    loc.longitude or 0)

df = df.reindex(columns=['FIPS_Combined', 'State', 'COUNTY',
    'latitude', 'longitude'])

df.to_csv('geocode.csv', index=False)
```

A.3 distance.py

```
import pandas as pd
import numpy as np
import geopy.distance
from tqdm import tqdm
import sklearn.metrics

tqdm.pandas()

df_geocode = pd.read_csv('geocode.csv')
df_geocode.set_index('FIPS_Combined', inplace=True,
    drop=False)

df_temp = df_geocode[['latitude', 'longitude']]

distance_matrix = sklearn.metrics.pairwise_distances(
    df_temp, metric=lambda a, b: geopy.distance.distance(a,
        b).miles, n_jobs=8)

df_distance = pd.DataFrame(distance_matrix,
                           columns=df_geocode['FIPS_Combined'],
                           index=df_geocode['FIPS_Combined'])

df_distance.to_csv('geocode_distance.csv')
```

A.4 count.py

```
import pandas as pd

df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
    sheet_name='Data')
df = df[['State', 'SubstanceName', 'DrugReports']]

df_grouped = df.groupby(['State',
    'SubstanceName'],as_index=False).count()

```

```
df_grouped.sort_values(by='DrugReports', inplace=True,
    ascending=False)
df_grouped.to_csv('drug_count.csv', index=False)
```

A.5 bpnn.py

```
import pandas as pd
import numpy as np
from tqdm import tqdm

total_years = 8
start_year = 2010

training_rate = 1e-6
weight_default = 0
weight_dim = 6
weight_dim_used = 6


class Path:
    def __init__(self, src, dest, dist):
        self.src = src
        self.dest = dest
        self.dist = dist
        self.weight = [weight_default for i in
            range(weight_dim)]


class County:
    def __init__(self, fips, name):
        self.fips = fips
        self.name = name
        self.paths = []
        self.train_data = [[0 for j in range(weight_dim)]
            for i in range(total_years)]
        self.predict_data = []
        self.aggregate_data = 0
        self.aggregate_data_size = [0 for i in
            range(weight_dim)]
        self.transfer_rate = 0

    def add_path(self, dest_county, dist):
        self.paths.append(Path(self, dest_county, dist))

    def set_train_data_drug(self, year, data):
        year = int(year)
        self.train_data[year][0] += data
        if year < total_years - 1:
            self.train_data[year + 1][1] += data

    def set_train_data_pca(self, year, data):
        year = int(year)
        for i, j in enumerate(data):
            self.train_data[year][i + 2] += j

    def prepare_train(self, year):
        self.aggregate_data = self.train_data[year + 1][0]
        self.aggregate_data_size = [0 for i in
            range(weight_dim)]

    def train(self, year):
        for path in self.paths:
            for i in range(2):
                if self.train_data[year][i] >= 10:
```

```
                        path.dest.aggregate_data -=
                            (path.weight[i] *
                            self.train_data[year][i])
                        path.dest.aggregate_data_size[i] += 1
                for i in range(2, weight_dim_used):
                    path.dest.aggregate_data -= (path.weight[i]
                        * self.train_data[year][i])
                    path.dest.aggregate_data_size[i] += 1

    def back_propagation(self, year):
        for path in self.paths:
            error = path.dest.aggregate_data /
                weight_dim_used
            for i in range(2):
                if self.train_data[year][i] > 10:
                    e = error /
                        path.dest.aggregate_data_size[i]
                    path.weight[i] = max(-1, path.weight[i]
                        + training_rate * e *
                        self.train_data[year][i])

            for i in range(2, weight_dim_used):
                e = error / path.dest.aggregate_data_size[i]
                path.weight[i] = max(-1, path.weight[i] +
                    training_rate * e *
                    self.train_data[year][i])

    def prepare_predict(self, years=10):
        self.predict_data = [[0 for j in range(weight_dim)]
            for i in range(years + 2)]
        self.predict_data[0] = self.train_data[-2]
        self.predict_data[1] = self.train_data[-1]

    def predict(self, year):
        for path in self.paths:
            for i in range(weight_dim_used):
                path.dest.predict_data[year + 2][0] +=
                    path.weight[i] *
                    self.predict_data[year][i]
        self.predict_data[year + 2][1] =
            self.predict_data[year + 1][0]
        for i in range(2, weight_dim):
            self.predict_data[year + 2][i] =
                self.predict_data[year + 1][i]


# df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
    sheet_name='Data')
df_distance = pd.read_csv('geocode_distance.csv',
    index_col=0)
df_pca = pd.read_csv('../result/pca.csv')


def get_distance(a, b):
    return df_distance[str(a)].loc[int(b)]


# state = 'OH'
# substance_name = 'Buprenorphine'


# state = 'KY'
# substance_name = 'Hydrocodone'


def bpnn(state, substance_name, n_components=0,
    reverse=False):
    global weight_dim_used
```

```
    weight_dim_used = 2 + n_components

    df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
        sheet_name='Data')
    df_county = df[df['State'] ==
        state].groupby('FIPS_Combined').first()
    df = df[(df['State'] == state) & (df['SubstanceName'] ==
        substance_name)].reset_index()

    county_list = dict()

    for fips, row in df_county.iterrows():
        county_list[fips] = County(fips, row['COUNTY'])

    max_min_distance = 0

    for _i, county_i in county_list.items():
        min_distance = float('inf')
        for _j, county_j in county_list.items():
            distance = get_distance(county_i.fips,
                county_j.fips)
            if distance and distance < min_distance:
                min_distance = distance
            if distance < 40:
                county_i.add_path(county_j, distance)
        # if max_min_distance < min_distance:
        #     print(_i, min_distance)
        max_min_distance = max(max_min_distance,
            min_distance)

    # print(max_min_distance)

    for index, row in df.iterrows():
        fips = row['FIPS_Combined']
        if reverse:
            year = start_year - 1 + total_years -
                row['YYYY']
        else:
            year = row['YYYY'] - start_year
        data = row['DrugReports']
        county_list[fips].set_train_data_drug(year, data)

    for index, row in df_pca.iterrows():
        fips = row['FIPS']
        if fips not in county_list:
            continue
        if reverse:
            year = start_year - 1 + total_years -
                row['YYYY']
        else:
            year = row['YYYY'] - start_year
        data = [row['D%d' % i] for i in range(1, 5)]
        county_list[fips].set_train_data_pca(year, data)

    mean = 0
    for year in range(total_years - 1):
        for _, county_i in county_list.items():
            mean += county_i.train_data[year + 1][0]
    mean = mean / (total_years - 1) / len(county_list)

    print(mean)

    u = 0
    error = 0
    for i in tqdm(range(1000)):
        # u = 0
        # v = 0
        # for year in range(total_years - 1):
```

```
            #     for _, county_i in county_list.items():
            #         county_i.prepare_train(year)
            #
            #     for _, county_i in county_list.items():
            #         county_i.train(year)
            #
            #     for _, county_i in county_list.items():
            #         u += county_i.aggregate_data ** 2
            #         v += (county_i.train_data[year + 1][0] - mean) ** 2
            # error = 1 - u / v
            # print(error, u)

            for year in range(total_years - 1):
                for _, county_i in county_list.items():
                    county_i.prepare_train(year)

                for _, county_i in county_list.items():
                    county_i.train(year)

                for _, county_i in county_list.items():
                    county_i.back_propagation(year)

        u = 0
        v = 0
        error = 0
        for year in range(total_years - 1):
            for _, county_i in county_list.items():
                county_i.prepare_train(year)

            for _, county_i in county_list.items():
                county_i.train(year)

            for _, county_i in county_list.items():
                u += county_i.aggregate_data ** 2
                v += (county_i.train_data[year + 1][0] - mean) ** 2
        error = 1 - u / v

        predict_years = 2

        for _, county_i in county_list.items():
            county_i.prepare_predict(predict_years)

        for i in range(predict_years):
            for _, county_i in county_list.items():
                county_i.predict(i)

        result_arr = []
        for _, county_i in county_list.items():
            # if county_i.predict_data[-1] >= 10:
            # print(_, county_i.train_data, county_i.predict_data)
            arr = [_]
            for i in range(total_years):
                arr.append(county_i.train_data[i][0])
            for i in range(2):
                arr.append(county_i.predict_data[2 + i][0])

            result_arr.append(arr)

        df_result = pd.DataFrame(result_arr)
        # df_result.set_index(0, inplace=True)

        df_result.to_csv('../result/bpnn_source_%s_%s.csv' % (state, substance_name), index=False)
        return [state, substance_name, n_components, error]
```

```
#
# arr = list()
#
# arr.append(bpnn('KY', 'Hydrocodone', 0, True))
# arr.append(bpnn('KY', 'Hydrocodone', 1, True))
# arr.append(bpnn('OH', 'Hydrocodone', 0, True))
# arr.append(bpnn('OH', 'Hydrocodone', 1, True))
# arr.append(bpnn('PA', 'Hydrocodone', 0, True))
# arr.append(bpnn('PA', 'Hydrocodone', 1, True))
# arr.append(bpnn('VA', 'Hydrocodone', 0, True))
# arr.append(bpnn('VA', 'Hydrocodone', 1, True))
# arr.append(bpnn('WV', 'Hydrocodone', 0, True))
# arr.append(bpnn('WV', 'Hydrocodone', 1, True))
#
# arr.append(bpnn('KY', 'Buprenorphine', 0, False))
# arr.append(bpnn('KY', 'Buprenorphine', 1, False))
# arr.append(bpnn('OH', 'Buprenorphine', 0, False))
# arr.append(bpnn('OH', 'Buprenorphine', 1, False))
# arr.append(bpnn('PA', 'Buprenorphine', 0, False))
# arr.append(bpnn('PA', 'Buprenorphine', 1, False))
# arr.append(bpnn('VA', 'Buprenorphine', 0, False))
# arr.append(bpnn('VA', 'Buprenorphine', 1, False))
# arr.append(bpnn('WV', 'Buprenorphine', 0, False))
# arr.append(bpnn('WV', 'Buprenorphine', 1, False))
#
# df_total = pd.DataFrame(arr, columns=['State',
    'Substance', 'N', 'R2'])
# df_total.to_csv('../result/bpnn.csv')


def bpnn_state(substance_name, reverse=False):
    global weight_dim_used
    weight_dim_used = 2
    df = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
        sheet_name='Data')
    df = df[df['SubstanceName'] ==
        substance_name].reset_index()
    df_state = df.groupby(['State', 'YYYY'],
        as_index=False).sum().reset_index()

    state_list = dict()

    for index, row in df_state.iterrows():
        state_name = row['State']
        state_list[state_name] = County(state_name,
            state_name)

    for _i, state_i in state_list.items():
        for _j, state_j in state_list.items():
            state_i.add_path(state_j, 0)

    for index, row in df_state.iterrows():
        state_name = row['State']
        if reverse:
            year = start_year - 1 + total_years -
                row['YYYY']
        else:
            year = row['YYYY'] - start_year
        data = row['DrugReports']
        state_list[state_name].set_train_data_drug(year,
            data)

    mean = 0
    for year in range(total_years - 1):
        for _, state_i in state_list.items():
            mean += state_i.train_data[year + 1][0]
    mean = mean / (total_years - 1) / len(state_list)

```

```
    print(mean)

    u = 0
    error = 0
    for i in tqdm(range(1000)):
        # u = 0
        # v = 0
        # for year in range(total_years - 1):
        #     for _, state_i in state_list.items():
        #         state_i.prepare_train(year)
        #
        #     for _, state_i in state_list.items():
        #         state_i.train(year)
        #
        #     for _, state_i in state_list.items():
        #         u += state_i.aggregate_data ** 2
        #         v += (state_i.train_data[year + 1][0] - mean) ** 2
        # error = 1 - u / v
        # print(error, u)

        for year in range(total_years - 1):
            for _, state_i in state_list.items():
                state_i.prepare_train(year)

            for _, state_i in state_list.items():
                state_i.train(year)

            for _, state_i in state_list.items():
                state_i.back_propagation(year)

    u = 0
    v = 0
    error = 0
    for year in range(total_years - 1):
        for _, state_i in state_list.items():
            state_i.prepare_train(year)

        for _, state_i in state_list.items():
            state_i.train(year)

        for _, state_i in state_list.items():
            u += state_i.aggregate_data ** 2
            v += (state_i.train_data[year + 1][0] - mean) ** 2
    error = 1 - u / v

    print(error)

    predict_years = 2

    for _, state_i in state_list.items():
        state_i.prepare_predict(predict_years)

    for i in range(predict_years):
        for _, state_i in state_list.items():
            state_i.predict(i)

    result_arr = []
    for _, state_i in state_list.items():
        # if county_i.predict_data[-1] >= 10:
        # print(_, county_i.train_data, county_i.predict_data)
        arr = [_]
        for i in range(total_years):
            arr.append(state_i.train_data[i][0])
        for i in range(2):
            arr.append(state_i.predict_data[2 + i][0])

```

```
            result_arr.append(arr)

        df_result = pd.DataFrame(result_arr)
        df_result.set_index(0, inplace=True)

        return df_result

        # df_result.to_csv('../result/bpnn_source_state_%s.csv'
            % substance_name, index=False)
        # return [substance_name, error]


def bpnn_state_bidirection(substance_name):
        df_result1 = bpnn_state(substance_name, False)
        df_result2 = bpnn_state(substance_name, True).drop([1,
            2, 3, 4, 5, 6, 7, 8], axis=1)
        df_result2.columns = ['a', 'b']
        df_result = pd.concat([df_result2[['b', 'a']],
            df_result1], axis=1)
        df_result.to_csv('../result/bpnn_source_state_%s.csv' %
            substance_name, index=False)
        return df_result


bpnn_state_bidirection('Buprenorphine')
bpnn_state_bidirection('Morphine')
```

A.6 bpnn_analysis.py

```
import pandas as pd

# state = 'OH'
# substance_name = 'Buprenorphine'
state = 'KY'
substance_name = 'Hydrocodone'
df = pd.read_csv('../result/bpnn_source_%s_%s.csv' % (state,
    substance_name))

Q1 = df.quantile(0.25)[10]
Q3 = df.quantile(0.75)[10]
IQR = Q3 - Q1
Low = Q1 - 1.5 * IQR
High = Q3 + 1.5 * IQR

df_high = df[df['10'] > High].copy()
df_high.sort_values('10', ascending=False, inplace=True)

df_county = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
    sheet_name='Data')
df_county =
    df_county.groupby('FIPS_Combined').first().reset_index()
df_county.set_index('FIPS_Combined', inplace=True)
df_county = df_county[['COUNTY', 'State']]

df_high.set_index('0', inplace=True)
df_high = df_high[['10']]

df_join = df_high.join(df_county)

for index, row in df_join.iterrows():
    print(row['COUNTY'], row['State'])

df_join.to_csv('../result/bpnn_source_%s_%s_county.csv' %
    (state, substance_name))
```

A.7 bpnn_plot.py

```
import plotly.figure_factory
import plotly

import pandas as pd
import numpy as np

plotly.tools.set_credentials_file(username='tc-imba',
    api_key='xNu6lsfY6Twz6LfUmjHa')

fips_dict = dict()

# state = 'OH'
# substance_name = 'Buprenorphine'
state = 'KY'
substance_name = 'Hydrocodone'
df = pd.read_csv('../result/bpnn_source_%s_%s.csv' % (state,
    substance_name))

colorscale = ["#deebf7", "#b3d2e9", "#6baed6", "#4292c6",
              "#2171b5", "#0b4083", "#08306b"]

for i in range(5):

    for index, row in df.iterrows():
        fips_dict[row[0]] = row[i + 1 + 4]
        # fips_dict[row[0]] = row[10 - i]

    fips = list(fips_dict.keys())
    values = list(fips_dict.values())

    endpts = list(np.geomspace(10, 200, 6))

    fig = plotly.figure_factory.create_choropleth(
        fips=fips, values=values,
        scope=[state],
        binning_endpoints=endpts,
        county_outline={'color': 'rgb(255,255,255)',
            'width': 0.5},
        legend_title='Drug Reports',
        colorscale=colorscale
    )

    fig['layout']['legend'].update({'x': 0.25, 'y': 0.75})
    fig['layout']['annotations'][0].update({'x': 0.12, 'y':
        0.8, 'xanchor': 'left'})

    # plotly.offline.plot(fig, filename='map.html')
    plotly.plotly.plot(fig, filename='%s_%s_%d' % (state,
        substance_name, i + 1), fileopt='overwrite',
        auto_open=True)
    # # plotly.plotly.image.save_as(fig, 'test.png',
        width=1920, height=1080)
```

A.8 bpnn_plot_stack.py

```
import pylab as pl
import numpy as np

from mayavi import mlab
from tvtk.api import tvtk

```

```
# state = 'OH'
# substance_name = 'Buprenorphine'
state = 'KY'
substance_name = 'Hydrocodone'


def draw_layer(path, position):
    # load a png with a scale 0->1 and four color channels (an extra alpha channel for transparency).
    im = pl.imread(path, format='png') * 255

    colors = tvtk.UnsignedCharArray()
    colors.from_array(im.transpose((1, 0, 2)).reshape(-1, 4))

    m_image = mlab.imshow(
        np.ones(im.shape[:2]),
        extent=[0, 0, 0, 0, position, position],
        opacity=0.6)

    m_image.actor.input.point_data.scalars = colors


if __name__ == "__main__":
    fig_num = 5
    fig_height = 200 * (fig_num - 1)

    mlab.figure(bgcolor=(0.97, 0.97, 0.97), size=(1000, 1000))

    for i in range(fig_num):
        draw_layer('../figure/%s_%s_%d.png' % (state, substance_name, i + 1), i * 200)

    im = pl.imread('../figure/%s_%s_1.png' % (state, substance_name), format='png')

    m_image = mlab.quiver3d(-im.shape[0] / 2, -im.shape[1] / 2, fig_height,
                            0, 0, -fig_height - 50,
                            line_width=.1, colormap='Blues',
                            scale_factor=1, mode='arrow',
                                resolution=25)

    m_image.glyph.glyph_source.glyph_source.shaft_radius = 0.005
    m_image.glyph.glyph_source.glyph_source.tip_length = 0.02
    m_image.glyph.glyph_source.glyph_source.tip_radius = 0.01

    mlab.view(azimuth=30, elevation=62)
    mlab.gcf().scene.parallel_projection = True
    mlab.gcf().scene.camera.zoom(1.1)

    mlab.draw()
    mlab.savefig('../figure/%s_%s.png' % (state, substance_name))
    mlab.show()
```

A.9 bpnn_plot_state.py

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

```

colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'grey']

substance_name = 'Buprenorphine'

df = pd.read_csv('../result/bpnn_source_state_%s.csv' %
    substance_name)


fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)
x = np.arange(2008, 2020)

for i, (name, row) in enumerate(df.iterrows()):
    color = colors[i]
    y = row.values
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.grid(True, 'both')
plt.xticks(x)
plt.xlabel('Year', fontsize=15)
plt.ylabel('Drug Report', fontsize=15)
plt.ylim([0, 2500])
plt.legend(loc='upper left', prop={'size': 15})
plt.title('Year vs Drug Report (%s)' % substance_name)
plt.savefig('../figure/state_%s.eps' % substance_name)
plt.show()


substance_name = 'Morphine'

df = pd.read_csv('../result/bpnn_source_state_%s.csv' %
    substance_name)


fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)
x = np.arange(2008, 2020)

for i, (name, row) in enumerate(df.iterrows()):
    color = colors[i]
    y = row.values
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.grid(True, 'both')
plt.xticks(x)
plt.xlabel('Year', fontsize=15)
plt.ylabel('Drug Report', fontsize=15)
plt.ylim([0, 600])
plt.legend(loc='upper right', prop={'size': 15})
plt.title('Year vs Drug Report (%s)' % substance_name)
plt.savefig('../figure/state_%s.eps' % substance_name)
plt.show()
```

A.10 bpnn_plot_transfer.py

```
import plotly.offline as py
import pandas as pd
```

```
3
4  df_airports = pd.read_csv(
5          'https://raw.githubusercontent.com/plotly/datasets/master/2011_
6  df_airports.head()
7
8  df_flight_paths = pd.read_csv(
9          'https://raw.githubusercontent.com/plotly/datasets/master/2011_
10 df_flight_paths.head()
11
12 airports = [dict(
13     type='scattergeo',
14     locationmode='USA-states',
15     lon=df_airports['long'],
16     lat=df_airports['lat'],
17     hoverinfo='text',
18     text=df_airports['airport'],
19     mode='markers',
20     marker=dict(
21         size=2,
22         color='rgb(255, 0, 0)',
23         line=dict(
24             width=3,
25             color='rgba(68, 68, 68, 0)'
26         )
27     ))]
28
29 flight_paths = []
30 for i in range(len(df_flight_paths)):
31     flight_paths.append(
32         dict(
33             type='scattergeo',
34             locationmode='USA-states',
35             lon=[df_flight_paths['start_lon'][i],
                     df_flight_paths['end_lon'][i]],
36             lat=[df_flight_paths['start_lat'][i],
                     df_flight_paths['end_lat'][i]],
37             mode='lines',
38             line=dict(
39                 width=1,
40                 color='red',
41             ),
42             opacity=float(df_flight_paths['cnt'][i]) /
                     float(df_flight_paths['cnt'].max()),
43         )
44     )
45
46 layout = dict(
47     title='Feb. 2011 American Airline flight paths<br>(Hover
          for airport names)',
48     showlegend=False,
49     geo=dict(
50         scope='north america',
51         projection=dict(type='azimuthal equal area'),
52         showland=True,
53         landcolor='rgb(243, 243, 243)',
54         countrycolor='rgb(204, 204, 204)',
55     ),
56 )
57
58 fig = dict(data=flight_paths + airports, layout=layout)
59 py.plot(fig, filename='d3-flight-paths')
```

A.11 PCA.py

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn import preprocessing

import numpy as np
from sklearn.linear_model import LinearRegression

df_data = pd.DataFrame()
for i in range(7):
    year = '1%d' % i
    filename =
        '../data/ACS_1%d_5YR_DP02/ACS_1%d_5YR_DP02_with_ann.csv'
        % (i, i)
    df_temp = pd.read_csv(filename, header=[0, 1])
    df_temp.columns = df_temp.columns.get_level_values(0)
    df_temp = df_temp.filter(regex='^(HC01|GEO.id2)',
        axis=1)
    columns = df_temp.columns.tolist()
    columns = ['YYYY', 'State'] + columns
    df_temp = df_temp.reindex(columns=columns)
    df_temp['YYYY'] = '201%d' % i
    df_temp['State'] = df_temp['GEO.id2'].apply(lambda fips:
        str(fips)[0:2])
    # df_temp['GEO.id2'] = df_temp['GEO.id2'].apply(lambda
        fips: '%d,201%d' % (fips, i))
    df_data = df_data.append(df_temp, sort=False)


def preprocess_data(x):
    if isinstance(x, str) and x == '(X)':
        return 0
    if not isinstance(x, str) and np.isnan(x):
        return 0
    return x


df_data = df_data.applymap(preprocess_data)
years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, None]
states = [21, 39, 42, 51, 54, None]
states_dict = {
    21: 'KY',
    39: 'OH',
    42: 'PA',
    51: 'VA',
    54: 'WV'
}


def apply_pca(state=None, year=None, n_components=10):
    df = df_data.copy()
    if state:
        df = df[df['State'] == str(state)]
    if year:
        df = df[df['YYYY'] == str(year)]
    df = df.reset_index(drop=True)

    X = df[df.columns[3:]].astype(np.float64)
    X = preprocessing.scale(X)

    pca = PCA(n_components=n_components)
    pca.fit(X)
    # print(pca.explained_variance_ratio_)

    A = pca.transform(X)

```

```
    df_result = pd.concat([df[df.columns[0:3]],
        pd.DataFrame(A)], axis=1)
    df_result['key'] = df_result['YYYY'].astype(str) +
        df_result['GEO.id2'].astype(str)
    df_result.set_index('key', inplace=True)
    df_result.drop(columns=['YYYY', 'State', 'GEO.id2'],
        inplace=True)

    df_test = pd.read_excel('../data/MCM_NFLIS_Data.xlsx',
        sheet_name='Data')
    df_test = df_test.groupby(['YYYY',
        'FIPS_Combined']).first().reset_index()
    # df_test = df_test[df_test['YYYY'] ==
        2010].reset_index()
    df_test['key'] = df_test['YYYY'].astype(str) +
        df_test['FIPS_Combined'].astype(str)
    df_test.set_index('key', inplace=True)
    df_test = df_test.filter(['TotalDrugReportsCounty'],
        axis=1)

    df_result = df_result.join(df_test)
    df_result.dropna(inplace=True)

    result_arr = []
    year_name = year and str(year) or 'All'
    state_name = state and states_dict[state] or 'All'

    for i in range(1, n_components + 1):
        X = df_result[df_result.columns[:i]].values
        y = df_result[df_result.columns[-1]].values

        data_length = X.shape[0]
        train_data_length = data_length * 2 // 3

        reg_score = []
        for j in range(10):
            arr = np.arange(data_length)
            np.random.shuffle(arr)

            train_X = X[arr[:train_data_length]]
            train_y = y[arr[:train_data_length]]
            test_X = X[arr[train_data_length:]]
            test_y = y[arr[train_data_length:]]

            reg = LinearRegression().fit(train_X, train_y)
            reg_score.append(reg.score(test_X, test_y))
            # print(reg.score(train_X, train_y))
            # print(reg.score(test_X, test_y))
            # print(reg.coef_)
            # print(reg.intercept_)

        explained_variance_ratio_sum =
            np.sum(pca.explained_variance_ratio_[:i])
        score = np.average(reg_score)

        result = [year_name, state_name, i,
            explained_variance_ratio_sum, score]
        result_arr.append(result)
        print(result)

    df = pd.DataFrame(result_arr, columns=['YYYY', 'State',
        'components', 'ratio', 'score'])
    return df, df_result


#
# df_plot = pd.DataFrame(columns=['YYYY', 'State',
    'components', 'ratio', 'score'])
```

```
# for i, state in enumerate(states):
#     df_temp, _ = apply_pca(state=state, n_components=10)
#     df_plot = df_plot.append(df_temp, ignore_index=True)
#
# df_plot.to_csv('../result/pca_state.csv', index=False)
#
# df_plot = pd.DataFrame(columns=['YYYY', 'State',
    'components', 'ratio', 'score'])
# for i, year in enumerate(years):
#     df_temp, _ = apply_pca(year=year, n_components=10)
#     df_plot = df_plot.append(df_temp, ignore_index=True)
#
# df_plot.to_csv('../result/pca_year.csv', index=False)
#
# df_plot = pd.DataFrame(columns=['YYYY', 'State',
    'components', 'ratio', 'score'])
# for i, state in enumerate(states):
#     df_temp, _ = apply_pca(state=state, year=2010,
    n_components=10)
#     df_plot = df_plot.append(df_temp, ignore_index=True)
#
# df_plot.to_csv('../result/pca_state_2010.csv',
    index=False)
#
# df_plot = pd.DataFrame(columns=['YYYY', 'State',
    'components', 'ratio', 'score'])
# for i, year in enumerate(years):
#     df_temp, _ = apply_pca(state=42, year=year,
    n_components=10)
#     df_plot = df_plot.append(df_temp, ignore_index=True)
#
# df_plot.to_csv('../result/pca_year_PA.csv', index=False)

df_PCA = pd.DataFrame(columns=['D1', 'D2', 'D3', 'D4',
    'YYYY', 'FIPS'])

for i in range(5):
    _, df_KY = apply_pca(state=states[i], n_components=4)
    df_KY = df_KY[[0, 1, 2, 3]]
    df_KY.rename(lambda x: 'D' + str(int(x) + 1), axis=1,
        inplace=True)
    df_KY['YYYY'] = df_KY.index.map(lambda x: x[0:4])
    df_KY['FIPS'] = df_KY.index.map(lambda x: x[4:])
    df_PCA = df_PCA.append(df_KY, ignore_index=True)

df_PCA.to_csv('../result/pca.csv', index=False)
```

A.12 PCA_plot.py

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

colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'grey']


def formatter_func(x, pos):
    return '%.1f' % x


df = pd.read_csv('../result/pca_state.csv')
n_components = df['components'].max()

```

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('State')):
    color = colors[i]
    x = group['components']
    y = group['ratio']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.yscale('log')
plt.grid(True, 'both')
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('explained variance ratio', fontsize=15)
plt.ylim([0.4, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs explained variance ratio')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_state_ratio.eps')
plt.show()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('State')):
    color = colors[i]
    x = group['components']
    y = group['score']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.grid(True)
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('coefficient of determination', fontsize=15)
plt.ylim([0, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs coefficient of determination')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_state_score.eps')
plt.show()

df = pd.read_csv('../result/pca_year.csv')
n_components = df['components'].max()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('YYYY')):
    color = colors[i]
    x = group['components']
    y = group['ratio']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.yscale('log')
```

```
plt.grid(True, 'both')
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('explained variance ratio', fontsize=15)
plt.ylim([0.75, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs explained variance ratio')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_year_ratio.eps')
plt.show()


fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('YYYY')):
    color = colors[i]
    x = group['components']
    y = group['score']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.grid(True)
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('coefficient of determination', fontsize=15)
plt.ylim([-1, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs coefficient of determination')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_year_score.eps')
plt.show()


df = pd.read_csv('../result/pca_state_2010.csv')
n_components = df['components'].max()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('State')):
    color = colors[i]
    x = group['components']
    y = group['ratio']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.yscale('log')
plt.grid(True, 'both')
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('explained variance ratio', fontsize=15)
plt.ylim([0.7, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs explained variance ratio
    (2010)')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_state_2010_ratio.eps')
plt.show()

```

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('State')):
    color = colors[i]
    x = group['components']
    y = group['score']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.grid(True)
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('coefficient of determination', fontsize=15)
plt.ylim([-1, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs coefficient of determination
    (2010)')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_state_2010_score.eps')
plt.show()

df = pd.read_csv('../result/pca_year_PA.csv')
n_components = df['components'].max()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('YYYY')):
    color = colors[i]
    x = group['components']
    y = group['ratio']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])

plt.yscale('log')
plt.grid(True, 'both')
plt.xticks(np.arange(1, n_components + 1))
plt.xlabel('components', fontsize=15)
plt.ylabel('explained variance ratio', fontsize=15)
plt.ylim([0.5, 1])
plt.legend(loc='lower right', prop={'size': 15})
plt.title('PCA components vs explained variance ratio (PA)')
ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
plt.savefig('../result/pca_year_PA_ratio.eps')
plt.show()


fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(1, 1, 1)

for i, (name, group) in enumerate(df.groupby('YYYY')):
    color = colors[i]
    x = group['components']
    y = group['score']
    h = plt.plot(x, y, label=name,
                 linestyle='-', linewidth=1,
                     color=colors[i],
                 marker='o', markersize=5,
                     markerfacecolor=colors[i])
```

```
203
204 plt.grid(True)
205 plt.xticks(np.arange(1, n_components + 1))
206 plt.xlabel('components', fontsize=15)
207 plt.ylabel('coefficient of determination', fontsize=15)
208 plt.ylim([-1, 1])
209 plt.legend(loc='lower right', prop={'size': 15})
210 plt.title('PCA components vs coefficient of determination
        (PA)')
211 ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatter_func))
212 ax.yaxis.set_minor_formatter(ticker.FuncFormatter(formatter_func))
213 plt.savefig('../result/pca_year_PA_score.eps')
214 plt.show()
```