STATS 5630 STATISTICAL MACHINE LEARNING

FINAL PROJECT

Analyzing the Prescription Habits of Doctors during the Opiate Overdose Epidemic

---

James Bonaffini
Max Ryoo
Abdou Karim Ndong

Introduction

With the growing rate of opioid overdose in the United States, our group decided to examine the Opioid crisis in the USA from the perspective of the prescribing doctors. We will be examining the prescribing habits of various doctors based upon their state, credentials, and number of drugs and opioids prescribed. In addition to this we will also be examining poor outcome rates in all states. We hope to be able to point to the possible predictors that could lead to poor outcomes such as doctor type, doctor gender, doctor location, or number of times a doctor prescribed opiates.

Data

The dataset that we are utilizing for this study is originally from the Centers for Medicare and Medicaid Services or CMS (cms.gov). CMS provides a wealth of data to support various research efforts, supply vital statistics about population health and status, and fulfill the compulsory transparency regulations that are built into publicly-owned entities.

Part of the data that is provided is all Medicare Part D prescription data (https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber). The data set that we used was drawn from the approximately 25 million records in the "Medicare Part D Prescriber Data CY 2014" by Alan Pryor, Ph.D. and uploaded to Kaggle. From these ~25 million records (each denoting a unique prescription instance), 25,000 were selected, cleaned, and compiled by Dr. Pryor. The predictors that were selected included the prescriber identification number, state, credentials, and specialty. Additionally, the number of specific prescriptions by drug name were listed as predictors.

Please see TABLE 1 for a list of the predictors. Displayed are some of the 256 different predictors, 250 of them being different drugs (only 28 are displayed). The ……. in the bottom row of the table denotes the remaining 222 predictor drugs that are not displayed in the table. NPI or National Provider Identifier number is the aforementioned prescriber identification number. Specialty is the type of medicinal practice. There is also an additional predictor "Opioid.Prescriber" that is a boolean value indicating whether the respective prescriber wrote opioid prescriptions more than 10 times in the 2014 calendar year.

TABLE 1		
NPI	Gender	State
Credentials	Specialty	ABILIFY
ACETAMINOPHEN.CODEINE	ACYCLOVIR	ADVAIR.DISKUS
AGGRENOX	ALENDRONATE.SODIUM	ALLOPURINOL
ALPRAZOLAM	AMIODARONE.HCL	AMITRIPTYLINE.HCL
AMLODIPINE.BESYLATE	AMLODIPINE.BESYLATE.BENAZEPRIL	AMOXICILLIN
AMOX.TR.POTASSIUM.CLAVULANATE	AMPHETAMINE.SALT.COMBO	ATENOLOL
ATORVASTATIN.CALCIUM	AVODART	AZITHROMYCIN
BACLOFEN	BD.ULTRA.FINE.PEN.NEEDLE	BENAZEPRIL.HCL
BENICAR	BENICAR.HCT	BENZTROPINE.MESYLATE
BISOPROLOL.HYDROCHLOROTHIAZIDE	BRIMONIDINE.TARTRATE	BUMETANIDE
.....		Opioid.Prescriber

Undoubtedly, there must be some processing on the dataset to make it more workable and easier to understand. Dr. Pryor did perform a moderate amount of the work by cleaning the dataset direct from CMS. There is a lot to be desired -- lots of missing, incorrect, or otherwise useless values. This is a lesson that for those that wish to integrate data across multiple platforms and perform some useful processing -- there is still much that must be done to get the data into a workable state.

Data Cleaning and Setup

The goal of doctor Dr. Pryor was to keep the dataset vague enough to be applied to multiple studies, thus a fair amount of further data analysis and cleaning had to be performed to get the dataset into state suitable for processing by machine learning algorithms. Specifically, we aimed to make it easier to analyze the data on the topic of opioids.

The first task was to clean the prescription or drug name fields into something more readable by data scientists. Thus, each drug was sorted into drug class as shown in TABLE 2. This research was performed by many Google searches. In the end, there were 62 different drug classes that were accounted for. Among the most numerous were opiates, anticonvulsants, corticosteroids, antidepressants, calcium channel blockers, beta blockers, and antibiotics. The 11 opiates that were accounted for in the dataset are displayed in TABLE 3. This is a subsection of the opiates available to the market. One assumption that we made was that Dr. Pryor had good reason to leave out the rest of the opiates. Dr. Pryor, according to his analysis on kaggle, simply gathered the most common opiates. Using the drug classifications, we created a mask, adding up the drug prescriptions to their respective classification, and replacing the 250 drug name predictors with these cleaned up 62 predictors.

More simpler cleaning tasks were also performed. This included encoding gender from M or F to one-hot of 2 different predictors Gender.M and Gender.F. For specialty, there was a similar one-hot encoding procedure that was formed, sorting practices into “surgery” and “internal medicine.” There are likely more categories that the specialties can be sorted into, and there were plenty of specialties that were left out of the two mentioned categories; however,

there were so many unique specialties with little in common that we decided to only include those two. Regarding credentials, there were many different varieties of clinicians. We decided to separate the prescribers into the 5 separate credentialed categories of "doctor," "nurse," "PA" (physician's assistant), "pharm" (pharmacy technician), and "other" based on the most advanced credential using one-hot encoding.

We also gathered data from 2014 about state populations and drug-related deaths.

TABLE 2					
antipsychotic	7	alpha.adrenergic.agonist	2	bronchodilator	3
opiate	11	diuretic	8	nitrate	3
antiviral	1	anticonvulsant	12	biguanide	3
corticosteroid	11	decarboxylase.inhibitor	1	DPP4 inhibitor	2
biphosphonate	2	anti.inflammatory	7	supplement	6
xanthine.oxidase.inhibitor	1	antimicrobial	1	laxative	2
benzodiazepine	6	anticoagulant	4	prostaglandin.analog	3
antiarrhythmic	1	antiplatelet	1	anesthetic	1
antidepressant	17	antifungal	3	antimetabolite	1
calcium.channel.blocker	11	antigout	1	prokinetic agent	1
antibiotic	12	proton.pump.inhibitor	5	leukotriene.receptor.antagonist	1
amphetamine	1	cardiac.glycoside	2	carboxylic acid	1
beta.blocker	12	antidiarrheal	1	NMDA.inhibitor	2
statin	8	carbonic anhydrase inhibitor	1	antiemetics	2
alpha.blocker	5	hormone	5	thiazolidinediones	1
muscle.relaxant	6	cholinesterase inhibitor	1	dopamine agonist	2
insulin	9	H2.blocker	2	anti.anginal	1
ACE.inhibitor	6	antilipemic	2	immunosuppressant	1
angiotensin.receptor.blocker	8	fibrate	1	protectant	1
anticholinergic	8	sulfonylurea	5	nucleoside analog	1
antihistimine	3	vasodilator	1		

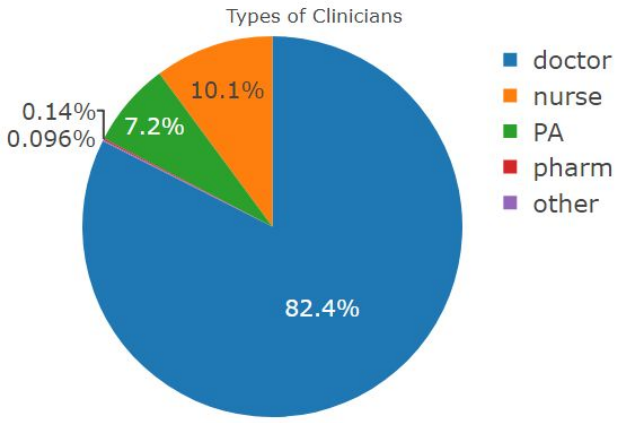
TABLE 3
ACETAMINOPHEN.CODEINE
FENTANYL
HYDROCODONE.ACETAMINOPHEN
HYDROMORPHONE.HCL
METHADONE.HCL
MORPHINE.SULFATE
MORPHINE.SULFATE.ER
OXYCODONE.ACETAMINOPHEN
OXYCODONE.HCL
OXYCONTIN
TRAMADOL.HCL

Initial Analysis

We performed an initial analysis on the data to see what descriptive information we could glean.

We first examined the clinician population. The vast majority of the clinicians were doctors, composing over 80% of the data's clinicians. Nurses and physician assistants composed a fair amount of the clinician population as well.

When it came to the prescription habits, we looked at some values of opiate prescriptions, including the top opioid prescribers mean and median amounts of opiates prescribed, and the total spread of opiate prescriptions. Examining TABLE 4, we found that mostly doctors from Pain Management, Anesthesiology, and Rehabilitation practices were prescribing the most amount of opiates. An eye-popping number of 15,234 opioid prescriptions were written by an Interventional Pain Management

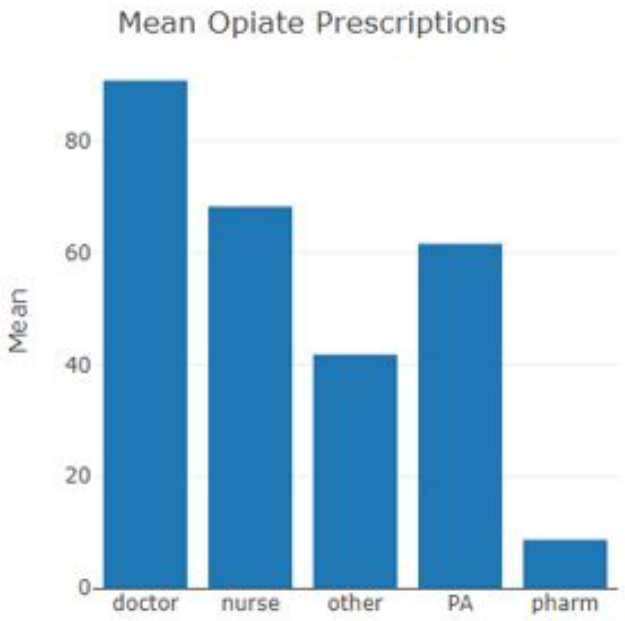


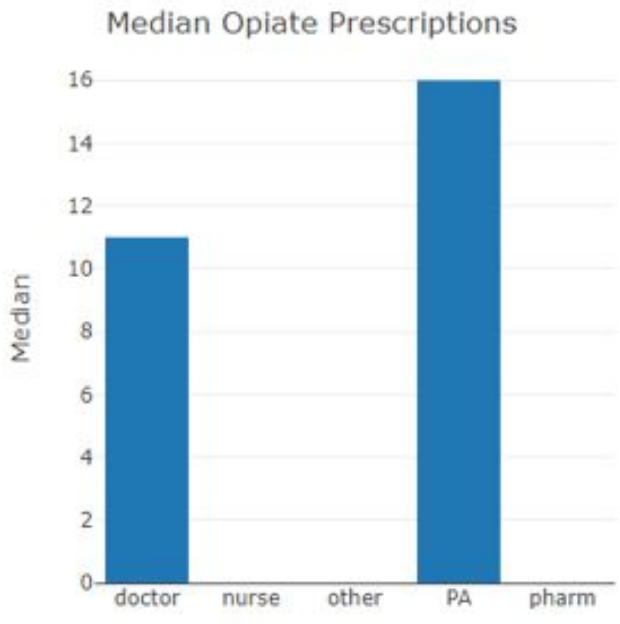
doctor in Florida. However, these values make some sense -- since opioids are intended as pain medications, Pain Management doctors would likely be giving out the most prescriptions. Without more in depth information as to the pill count or specifics about the patient, it is difficult to assume whether more prescriptions equals worse outcomes. Perhaps more prescriptions are good if the pill count is low, as the doctor is not giving out large, multi-month-long prescriptions.

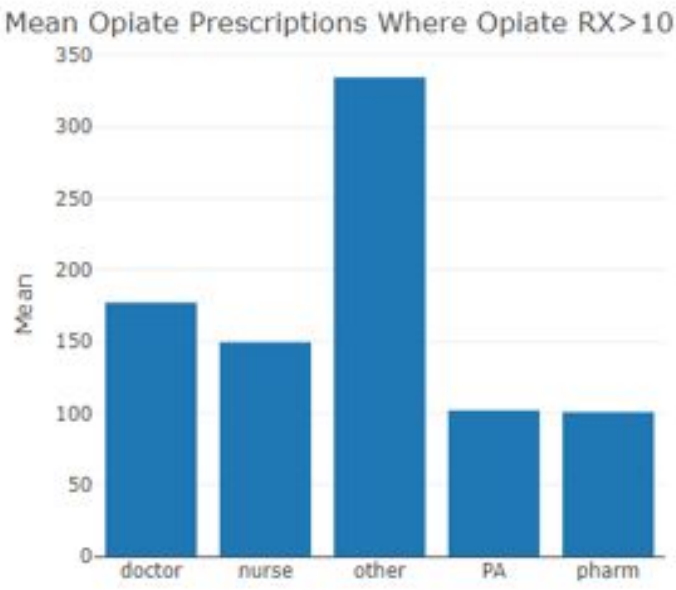
The average doctor is prescribing in the neighborhood of 90 opioid prescriptions per year - examine the graph labeled "Mean Opiate Prescriptions." If only looking at doctors above 10 opioid prescriptions per year, that number doubles. If examining the median values, that value for doctors increases 6-fold (see "Median Opiate Prescriptions" and "Median Opiate Prescriptions Where Opiate RX>10"). Nurses and PAs prescribed less opiates than doctors in nearly all cases over the course of 2014.

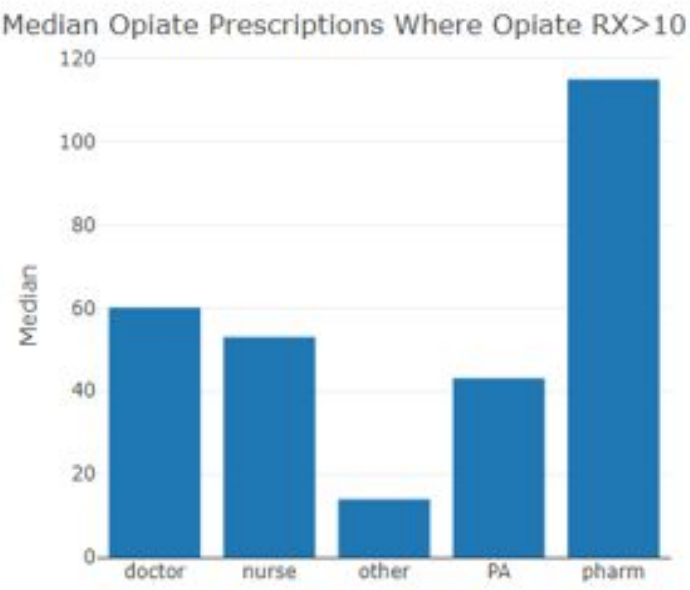
TABLE 4: Top 20 Opioid Prescribers

	State	Credentials	Specialty	opiate
1	FL	M.D.	Interventional Pain Management	15234
2	LA	MD	Physical Medicine and Rehabilitation	14674
3	LA	M.D.	Interventional Pain Management	6003
4	IN	MD	Pain Management	5890
5	FL	DO	Anesthesiology	5793
6	FL	MD	Anesthesiology	5021
7	MS	MD	Family Practice	4941
8	OH	MD	Pain Management	4687
9	NC	M.D.	Pain Management	4473
10	AR	MD	Pain Management	4283
11	FL	MD	Physical Medicine and Rehabilitation	4271
12	FL	M.D.	Physical Medicine and Rehabilitation	4013
13	OK	M.D.	Anesthesiology	3958
14	NC	FNP	Nurse Practitioner	3893
15	KS	M.D.	Anesthesiology	3691
16	OK	MD	Pain Management	3665
17	TX	DO	Physical Medicine and Rehabilitation	3634
18	NY	RPA-C	Physician Assistant	3591
19	WA	ARNP	Nurse Practitioner	3573
20	AZ	M.D.	Pain Management	3547









Next we looked at the overdose data by state. When we initially started this investigation, **we had hypothesized that more opiate prescriptions equaled more deaths**. However this table shows otherwise. The states with the most opiate prescriptions, mostly found in the southeast US, did not have the most deaths when ranked against other states, as shown in TABLE 5. States ranked by the number of opiates prescribed showed similar results. There also **seems to be very little correlation between the opiates prescribed and the number of deaths per 100,000 people in the state**. Two states, New Mexico and New Hampshire, were in the bottom 20% of average opiates prescribed, yet they had the number 2 and 3 highest deaths per

TABLE 5: States Ranked by Avg/Med Opiates Prescribed(Upper/Mid) and Deaths/100K(Lower)

Population	Deaths	State	medOpiate	avgOpiate	deathper100k	rank.avgOpiate	rank.medOpiate	rank.deaths
4,833,722	723	AL	26	178.37	14.96	1	2	25
4,625,470	777	LA	13	170.64	16.80	2	23	23
2,991,207	336	MS	33	154.16	11.23	3	1	43
2,959,373	356	AR	22	142.69	12.03	4	5	38
6,495,978	1,269	TN	17	137.61	19.54	5	12	11

Population	Deaths	State	medOpiate	avgOpiate	deathper100k	rank.avgOpiate	rank.medOpiate	rank.deaths
2,991,207	336	MS	33	154.16	11.23	3	1	43
4,833,722	723	AL	26	178.37	14.96	1	2	25
1,015,165	125	MT	24	92.92	12.31	20	3	36
2,959,373	356	AR	22	142.69	12.03	4	5	38
1,612,136	212	ID	22	90.90	13.15	22	5	35

Population	Deaths	State	medOpiate	avgOpiate	deathper100k	rank.avgOpiate	rank.medOpiate	rank.deaths
1,854,304	627	WV	14	99.21	33.81	18	18.0	1
2,085,287	547	NM	0	59.52	26.23	43	43.5	2
1,323,459	334	NH	11	40.61	25.24	46	34.0	3
4,395,295	1,077	KY	0	110.93	24.50	13	43.5	4
11,570,808	2,744	OH	11	92.42	23.71	21	34.0	5

100,000, respectively. This led us to believe that the number of deaths could not be predicted based on opiate prescription data. However, we had collated lots of other prescription data as well, so we decided to perform some regression analysis as well as employ unsupervised approaches to see if there were other patterns in the data that we could not see based on our initial analysis.

Machine Learning Methods and Results

Supervised: Regression and KNN

With the cleaned dataset, we tried to predict the number of deaths per 100,000 people per state. 80% of the data was used as the training set, reserving the final 20% for testing. There were a total of 77 predictors (most of the predictors were different drug classes) and 24680 observations (after missing values had been purged).

TABLE 6

	MSE
Linear Reg	20.37189
AIC Step	20.38088
AIC Forward	20.38331
BIC Step	20.41889
BIC Forward	20.40971
Ridge Reg	20.39705
Lasso Reg	20.39326

We found that, in accordance with what was predicted in our initial analysis, there was little predictive ability between the number of deaths and the amount of opioids prescribed -- MSEs were about 20, which is not great considering the maximum value is ~34. There was not much variation between the different methods of regression that we used; this was due to the fact that the intercept value dominated. Furthermore, over 95% of predicted values were between 14 and 16 deaths per 100,000 people. Of the chosen predictors from AIC and BIC, the antipsychotic, antidepressant, the angiotensin receptor blocker classes contributed the most to the prediction. Overall, this is not a useful form of prediction.

TABLE 7

	MSE
k=10	22.06087
k=20	21.04554
k=30	20.81493
k=40	20.6957
k=50	20.6161

KNN had similar results, with the MSE leveling out at about 20.5. The predicted values were similarly spread, with 80% of the values between 14 and 16 deaths per 100,000 people.

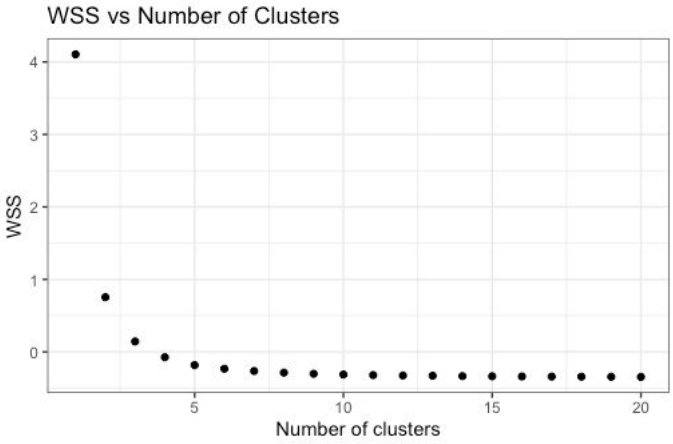
Abridged datasets with less or constrained predictors were also used, producing similar results. Some of these datasets included:

- Only specialties that appeared in the top 20 prescribers
- Only prescribers that prescribed less than 500 opioid prescriptions (to nullify outliers) or greater than 0 opioid prescriptions (to nullify prescribers who did not prescribe opioids).
- Only prescribers whose states were in to top 5 deaths per 100,000 people (MSE was lower, but this was because there was less variation in the response. There was similarly no variation in the predicted values).

These supervised learning approaches do not introduce enough variation in their predicted values and level off in their predictive ability at about 20 mean squared error. Unsupervised learning may be a better way to find patterns in the data for the purpose of better prediction of death in the United States during the opioid epidemic.

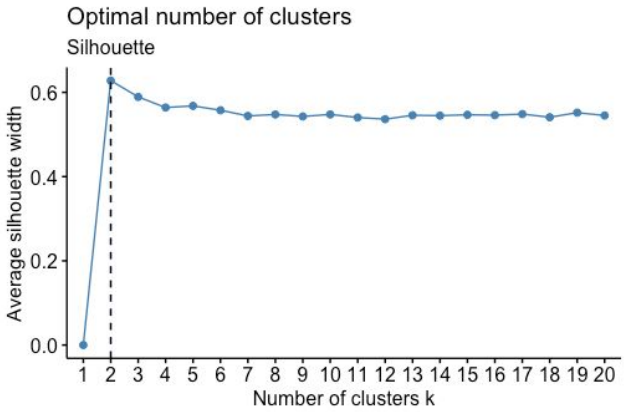
Clustering

With the clean dataset with all the attributes of the dataset as columns, unsupervised learning was conducted using clustering methods to gain insight into how many true clusters there were in the dataset. For the scope of this study, the clustering method of K-means clustering was used to find the true number of clusters in the dataset. To analyze how many true clusters were in the dataset, total centroids of 1 to 30 centroids were explored. The graph below shows the total within-cluster sum of square (WSS) over the number of centroids. Within-cluster sum of square is defined as the total distance of data points from their respective cluster centroids.



As one can see from the graph after around 4 - 5 clusters, the WSS does not change as much compared to the 4 or 5 clusters. From this graph alone one can argue that the true clusters of the dataset can be around 3 or 4 clusters. However, WSS is not the only way to investigate how many true clusters there are. Along with the WSS method the average silhouette function, and Gap statistics method was investigated.

The average silhouette method measures the quality of a certain clustering algorithm. The average silhouette method computes the average silhouette of observations from different values of k clusters, which in this case was from 1 to 20 clusters. The optimal clusters for this case would be the number of clusters that maximizes the average silhouette value. The following graph is shown for running the average silhouette method on the full dataset.

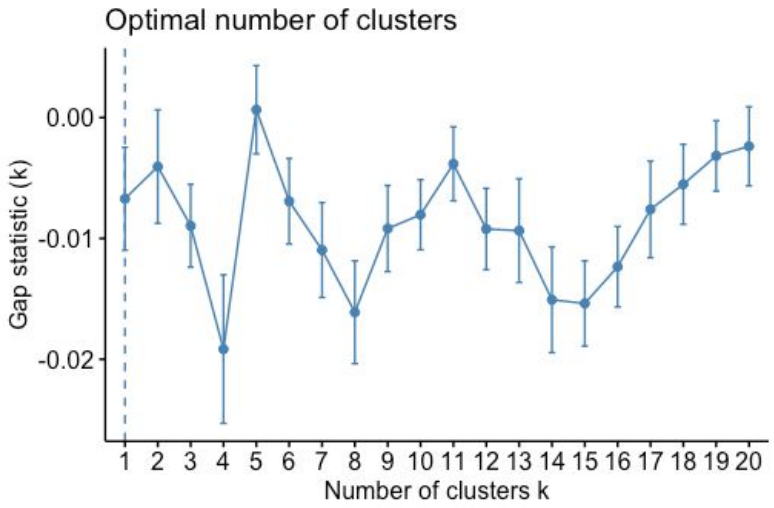


The graph shows that the optimal number of clusters is two since it maximizes the average silhouette values. After inspecting the results of the average silhouette method, the Gap Statistic method was investigated to see if there was a difference in finding the true number of

clusters. The gap statistics compares the total intracluster variation for different values of k with their expected values under null reference distribution of the data. The following equation defines gap statistic.

$$Gap_n(k) = E_n^* log(W_k) - log(W_k)$$

The number of k clusters that were looked at for the study was between 1 and 20 clusters like all other methods. The gap statistic method also required a number of bootstraps that was set to 20 bootstrap samples. The graph below shows the corresponding gap statistics for different values of k clusters.



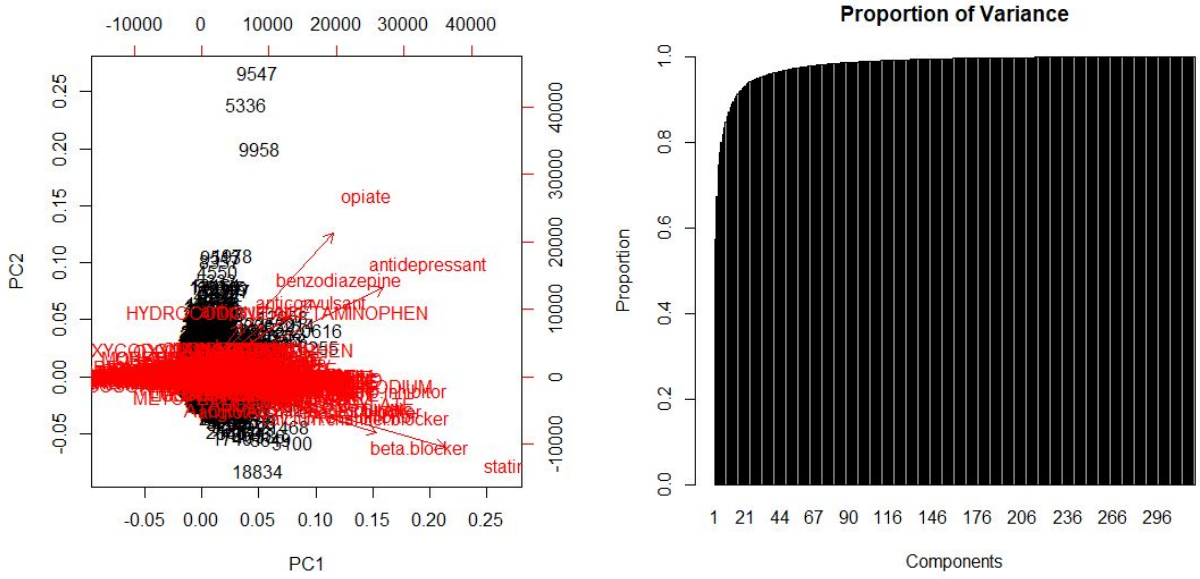
The graph that shows the gap statistic is the greatest at the k value of 5. Therefore using the gap statistics evaluation method the optimal number of clusters is 5. Since the dataset we are investigating doesn't have a known cluster, there is no correct number of clusters for this dataset. However, from previous evaluation methods cluster of 4 seems to be a middle ground. In a more broad spectrum one can say that the number of clusters for this opiate dataset is between 2 and 5.

Principal Component Analysis

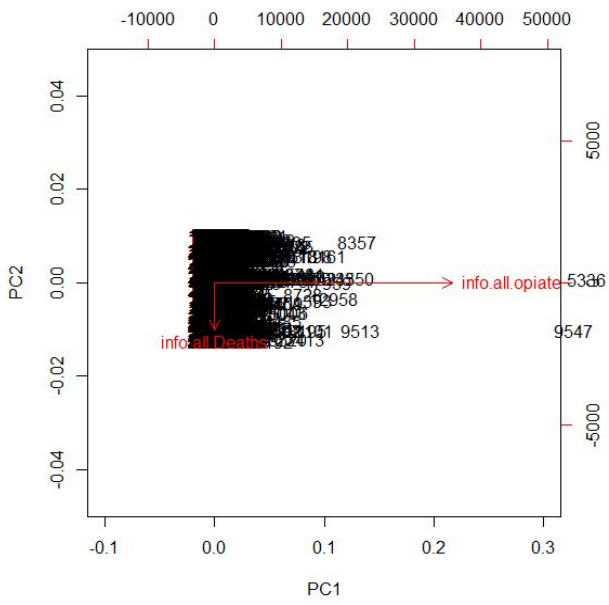
Principal component analysis is a statistical technique that is useful in identifying outlying units, across multiple variables contained in different dimensions. The approach is designed to convert numerous correlated variables into a smaller number of uncorrelated components, where the first component explains the maximum amount of variation between the original variables as possible. The approach is useful in detecting outliers in multivariate data sets by condensing the number of dimensions across multiple variables into a smaller set of data, without losing important information on the underlying variation. To identify possible outliers across the states with respect to the four measures analyzed, two individual PCA models were fit to the data generated. Before conducting the PCA, each of the input variables drug related

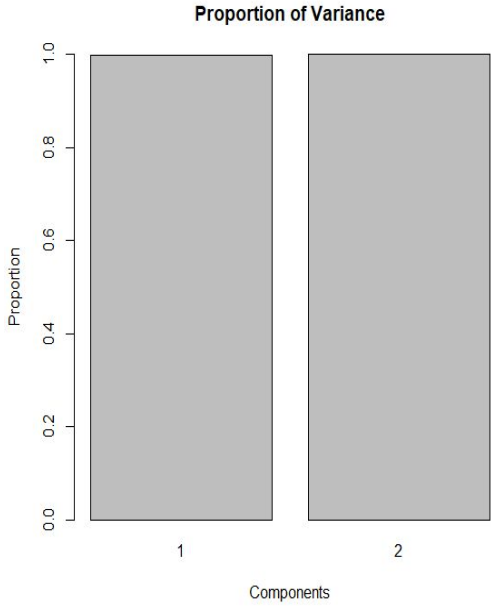
mortality rates and opiates prescription were standardized. The principal component was applied to dataset to identify potential opioid dependence risk at the country level.

As highlighted in the right Figure below, only three outliers could be identified from the datasets and simple cases allow the inspection of PC1 vs PC2. Thus, all the variables in the dataset are used for the PCA and the number of components that account for a variance of 90% are 14 and that is shown in the left Figure below. In other words 14 variables only variables are accounted for a variance of 90% and they are the following ones in the ascending order: H2.blocker, anticholinergic, xanthine.oxidase.inhibitor, sulfonylurea, angiotensin.receptor.blocker, hormone, biguanide, supplement, beta.blocker, proton.pump.inhibitor, ACE.inhibitor, statin, calcium.channel.blocker, diuretic. This means that this dataset which involves drugs used by patients could be narrowed down to these 14 drugs or prescriptions without losing information in the dataset.



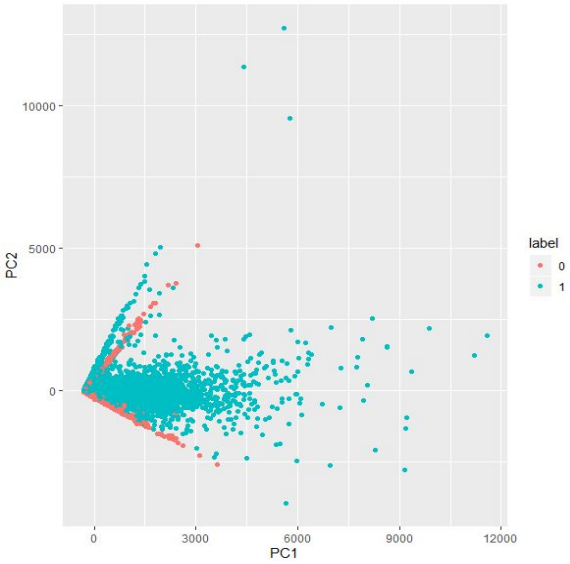
The relationship between the opiates and deaths will be investigated by setting the data frame between these two variables in order to see if there is any correlation between them. Figure below on the left shows that the two variables form a 90 degree angle, thus the two variables are uncorrelated. The number of opiate prescriptions is uncorrelated with the number of deaths that occurs across the country and that is shown in the left Figure below. Proportion of variance which is a generic term to mean a part of variance as a whole, reveals that the total variance between deaths and Opiate prescription is 100% each, each of which have their own proportion associated with them.





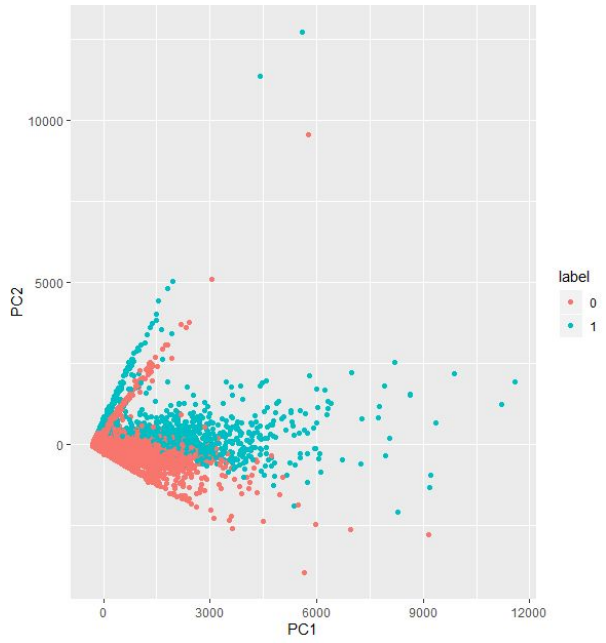
We have created binary variables to see the effect of the opiates prescription. The binaries involve of the opiate prescriptions of less than 10 and opiate prescriptions of more than 10, which stands for 1 and 0, respectively. The same is done with the prescriptions of 50 and less, 500 and less, and 1000 and less. It can be seen that the opiates prescriptions of 10 or less than will have both PC1 and PC2 of the two created binary variables matching and changes the PC1 as we increase the number of prescriptions. From these results, it can be inferred that there is a pattern in our dataset of drugs. Grouping the data in a certain way like clustering may reveal some new patterns in the data.

Opiates prescription (Less than 10 for 1 or more than 10 for 0)

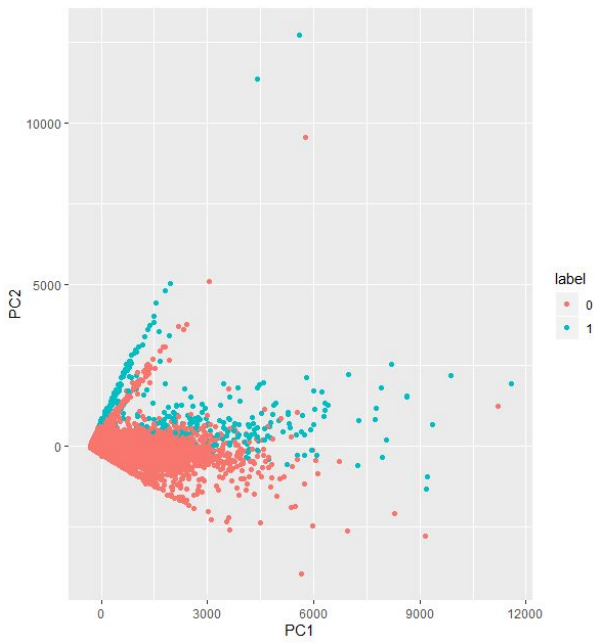


Opiates prescription (Less than 50 for 1 or more more than 50 for 0)

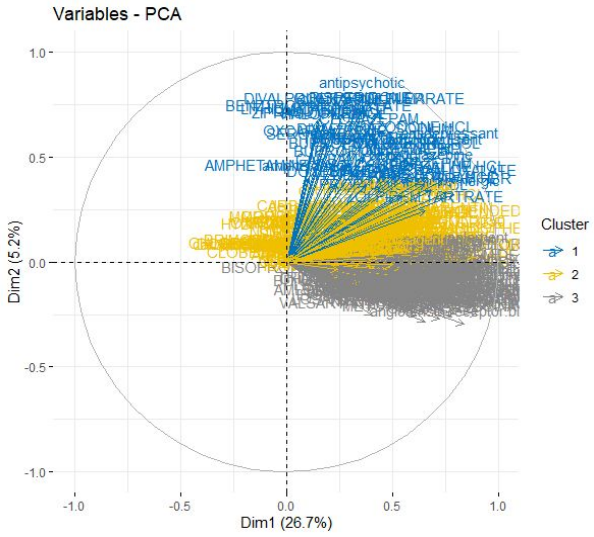
Opiates prescription (Less than 500 for 1 or more more than 500 for 0)



Opiates prescription (Less than 1000 for 1 or more more than 1000 for 0)



Next, we will apply the Principal Component Analysis and K-means Clustering to our dataset. This is dimensionality reduction by PCA and k-means clustering is used to visualize patterns in the dataset of drugs only. In this step, we will use k-means clustering to view the top three PCA components. In order to do this, we will first fit the computed principal components to the k-means algorithm and visualize the best number of clusters needed. Determining the ideal number of clusters for our k-means model can be done by measuring the sum of the squared distances to the nearest cluster center aka inertia. Classifying the drug variables into 3 groups using k-means clustering algorithm results that there is a pattern in our data though the variance between the cluster seem not to be significant.



Conclusion

The data that was used was readily accessible from Centers for Medicare and Medicaid Services or CMS. However, in order to do analysis the data needed lots of work to be done in order to clean and interpret the data. Upon data cleaning, many models could be made, but the generalization of these models to the problem and dataset were not very successful. For supervised learning methods Regression and KNN were investigated. Both of the models were not successful at predicting the impact of opiates on drug-related deaths. For the unsupervised learning methods Clustering and PCA were investigated, which was able to give more insight into the dataset in comparison to the supervised learning algorithms.

However, the two methods, supervised and unsupervised, had their own pros and cons in regard to analysis. Supervised learning has some pros of being able to easily see the input and output of the algorithm. Regardless of how good or bad the algorithm was, the weights of factors and indicators were very clear to see, which was not so prevalent in Unsupervised learning algorithms. Unsupervised learning gave more insight into the patterns of the data. Although the patterns were very subtle, it was able to give us insight into possibilities that were within the data. However, in Unsupervised learning it was hard to understand how the clusters were clustered and it was hard to directly distinguish which factor/factors were the cause of drug-related overdoses.

While investigating, the thought that many of opiate related deaths could be due to illegally obtained drugs was surfaced, which was not included in the dataset. A future direction that could help with the analysis of this epidemic will be to incorporate illegal drug transactions through different forces such as the police force. Also, we could apply these model concepts to pharmaceutical manufacturing data. If this is done we could possibly gain more insight into the illegal opiate substance abuse in comparison to a prescription.

There wasn't a model that easily predicted the drug overdose level nor gave very clear insight. However, many new lessons were learned, from the beginning of collection of data to the post analysis of the data. The opiate epidemic is a very serious topic and much more investigation and analysis is needed. We believe the topic is worth exploring further and more extensively, as it may be able to point to the source of the epidemic. This could, in the future, save lives, by informing policy-makers, doctors, and patients regarding the real reasons for drug-related deaths, permitting them to take impactful action.

```
##### ALL CODE
# Final Project
# Data Setup

# Packages
# install.packages("hashmap")
# install.packages("plotly")
# install.packages("plyr")
# install.packages("mltools")

# Libraries
library(hashmap)
library(plyr)
library(data.table)
library(mltools)

# import data from file
setwd("c:/Users/james/Data/UVA_ME/UVA_FALL_2019/STAT5630_StatisticalML/Project/")
opioids = read.csv("data/opioids.csv")
info = read.csv("data/prescriber-info.csv")
overdose = read.csv("data/overdoses.csv")
class = read.csv("data/drugclass.csv")
cred = read.csv("data/credentials.csv")
spec = read.csv("data/specialty.csv")
colnames(class)[1] = "Drug.Name"

# add drug class to the info matrix
classunique=unique(class$Drug.Class)
info.drug = colnames(info[,6:255])
classmask = data.frame(row.names=classunique)

for (c in 1 : length(classunique)) {
  r = info[1,6:255]
  rownames(r) = classunique[c]
  for (k in 1:length(info.drug)) {
    if (info.drug[k] == class$Drug.Name[k] && class$Drug.Class[k] == classunique[c]){
      r[k] = 1
    } else {
      r[k] = 0
    }
  }
  classmask = rbind(classmask,r)
}

# Apply mask
info.all = cbind(info,data.frame(t(as.matrix(classmask) %*% as.matrix(t(info[,6:255])))))

# Clean up credentials - add in rows
credunique=unique(cred$type)
credh = hashmap(cred$credential,cred$type)
credmat = as.data.frame(matrix(0,ncol=length(levels(credunique)),nrow=nrow(info.all)))
colnames(credmat) = levels(credunique)
for (k in 1:nrow(info.all)) {
  credmat[k,credh[[info.all$Credentials[k]]]]=1
}

info.all = cbind(info.all[,1:4],credmat,info.all[,5:length(info.all)])

# Clean up Specialty
specmat = as.data.frame(matrix(0,ncol=ncol(spec)-1,nrow=nrow(info.all)))
colnames(specmat) = colnames(spec)[2:ncol(spec)]

for (i in 1:(ncol(spec)-1)) {
  spech = hashmap(spec$Specialty,spec[,i+1])
  for (k in 1:nrow(info.all)) {
    specmat[k,i]=spech[[info.all$Specialty[k]]]
  }
}

info.all = cbind(info.all[,1:10],specmat,info.all[,11:length(info.all)])

# Add in death rates
OD.state = overdose[,2:4]
colnames(OD.state) <- c("Population", "Deaths", "State")

info.all <- merge(info.all, OD.state)
# info.all$deathrate <- as.integer(info.all$Deaths) / as.integer(info.all$Population)
info.all$deathrate <- as.numeric(gsub(",","",info.all$Deaths,fixed=TRUE)) /
as.numeric(gsub(",","",info.all$Population,fixed=TRUE))
```

```
info.all$deathrate = info.all$deathrate/max(info.all$deathrate)
info.all$deathper100k = round(as.numeric(gsub(",","",info.all$Deaths,fixed=TRUE)) /
(as.numeric(gsub(",","",info.all$Population)) / 100000),2)


# Convert M to 1 and F to 2
gender = one_hot(as.data.table(info.all$Gender))
colnames(gender)=c("Gender.F","Gender.M")
info.all = cbind(info.all[,1:2],gender,info.all[,4:length(info.all)])

info.all.back = info.all

# If you want to work with JUST numbers, use these dataframes
# info.all has ALL drug data and drug classification data
# info.class has just drug classification data

info.all$Specialty = NULL
info.all$Credentials = NULL
info.class = cbind(info.all[,1:10],info.all[,262:length(info.all)])

info.percent = info.class
info.percent$sums = rowSums(info.class[,12:73])
for (i in 12:73) {
  info.percent[,i] = as.numeric(info.percent[,i]) / as.numeric(info.percent$sums)
}

# Remember that we still need to separate into testing and training data
# For our response, we should use either the number of deaths or the deathrate per state
# Please note that our data is still not totally clean -- there are some nonsense states
#   Look at the Dataviz script for more info
```

```
#####
# Data Description/viz

# Packages
# install.packages("plotly")

# libraries
library(plotly)

# Doctors by state
states = unique(info.all$State)
doc.states = data.frame(states=matrix(0,ncol=1,nrow=length(states)))
rownames(doc.states) = states
for (i in 1:length(states)) {
  doc.states[i,1]=nrow(info.all[info.all$State==states[i],])
}

plot_ly(x=rownames(doc.states),
        y=doc.states$states, type = 'bar') %>%
  layout(title = 'Number of Clinicians by State',
         yaxis = list(title="Number of Clinicians"),xaxis = list(title="State"))

# Types of clinicians

clin_tot <- data.frame(observation= colnames(credmat),
                       value = c(colSums(credmat)))
plot_ly(data=clin_tot, labels = ~observation, values = ~value, type = 'pie',
        textfont = list(size = 20)) %>%
  layout(title = 'Types of Clinicians',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         legend = list(font = list(size=22)))

# Clinicians that give at least 10 prescriptions of opiates
clin_opiate = cbind(info.all[,5:9],info.all$Opioid.Prescriber,info.all$opiate)
colnames(clin_opiate) = c("doctor","nurse","other","PA","pharm","Opioid.Prescriber","opiate")
clin_opiate=clin_opiate[clin_opiate$Opioid.Prescriber==1,]

clin_tot_opiate <- data.frame(observation= colnames(credmat),
                              value = c(colSums(clin_opiate[,1:5])))
plot_ly(data=clin_tot_opiate, labels = ~observation, values = ~value, type = 'pie',
        textfont = list(size = 20)) %>%
  layout(title = 'Types of Clinicians With > 10 Opioid Prescriptions',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         legend = list(font = list(size=22)))

# Average Prescriptions Given out by each type of Clinician
clin_opiate2 = cbind(info.all[,5:9],info.all$opiate)
colnames(clin_opiate2) = c("doctor","nurse","other","PA","pharm","opiate")
clin_opiate_morethan10 = clin_opiate2[clin_opiate2$opiate>10,]

clin_tot_opiate_mean=c(mean(clin_opiate2$opiate[clin_opiate2[1]==1]),
                       mean(clin_opiate2$opiate[clin_opiate2[2]==1]),
                       mean(clin_opiate2$opiate[clin_opiate2[3]==1]),
                       mean(clin_opiate2$opiate[clin_opiate2[4]==1]),
                       mean(clin_opiate2$opiate[clin_opiate2[5]==1]))

clin_tot_opiate_mean_morethan10=c(mean(clin_opiate_morethan10$opiate[clin_opiate_morethan10[1]==1]),
                                  mean(clin_opiate_morethan10$opiate[clin_opiate_morethan10[2]==1]),
                                  mean(clin_opiate_morethan10$opiate[clin_opiate_morethan10[3]==1]),
                                  mean(clin_opiate_morethan10$opiate[clin_opiate_morethan10[4]==1]),
                                  mean(clin_opiate_morethan10$opiate[clin_opiate_morethan10[5]==1]))

clin_tot_opiate_median=c(median(clin_opiate2$opiate[clin_opiate2[1]==1]),
                         median(clin_opiate2$opiate[clin_opiate2[2]==1]),
                         median(clin_opiate2$opiate[clin_opiate2[3]==1]),
                         median(clin_opiate2$opiate[clin_opiate2[4]==1]),
                         median(clin_opiate2$opiate[clin_opiate2[5]==1]))

clin_tot_opiate_median_morethan10=c(median(clin_opiate_morethan10$opiate[clin_opiate_morethan10[1]==1]),
                                    median(clin_opiate_morethan10$opiate[clin_opiate_morethan10[2]==1]),
                                    median(clin_opiate_morethan10$opiate[clin_opiate_morethan10[3]==1]),
                                    median(clin_opiate_morethan10$opiate[clin_opiate_morethan10[4]==1]),
                                    median(clin_opiate_morethan10$opiate[clin_opiate_morethan10[5]==1]))


plot_ly(x=c("doctor","nurse","other","PA","pharm"),
             y=clin_tot_opiate_mean, type = 'bar') %>%
```

```
  layout(title = 'Mean Opiate Prescriptions',
         yaxis = list(title="Mean"))

plot_ly(x=c("doctor","nurse","other","PA","pharm"),
              y=clin_tot_opiate_mean_morethan10, type = 'bar') %>%
  layout(title = 'Mean Opiate Prescriptions Where Opiate RX>10',
         yaxis = list(title="Mean"))

plot_ly(x=c("doctor","nurse","other","PA","pharm"),
              y=clin_tot_opiate_median, type = 'bar') %>%
  layout(title = 'Median Opiate Prescriptions',
         yaxis = list(title="Median"))

plot_ly(x=c("doctor","nurse","other","PA","pharm"),
              y=clin_tot_opiate_median_morethan10, type = 'bar') %>%
  layout(title = 'Median Opiate Prescriptions Where Opiate RX>10',
         yaxis = list(title="Median"))

# p <- subplot(p1, p2, p3, p4,nrows = 2, ncols=2)
# p

# Histogram

plot_ly(x=clin_opiate$opiate[clin_opiate$opiate>1 & clin_opiate$opiate<5000],
        type = 'histogram',nbinsx=20) %>%
  layout(title = 'Number of Opiate Prescriptions Where Opiate 1<RX<5000',
         yaxis = list(type='log',title="Number of Prescriptions (log)"),
         xaxis = list(title="Clinicians (binned)"))

plot_ly(x=clin_opiate$opiate[clin_opiate$opiate>1 & clin_opiate$opiate<300],
        type = 'histogram',nbinsx=20) %>%
  layout(title = 'Number of Opiate Prescriptions Where Opiate 1<RX<500',
         yaxis = list(type='log',title="Number of Prescriptions (log)"),
         xaxis = list(title="Clinicians (binned)"))

# Other Statistics
colMax <- function(data) sapply(data,max,na.rm = TRUE)


# Clinicians for top # of prescriptions
clin_opiate = cbind(info.all.back[,1:11],info.all$Opioid.Prescriber,info.all$opiate)
colnames(clin_opiate) =
c("State","NPI","Gender.F","Gender.M","Credentials","doctor","nurse","other","PA","pharm","Specialty","Opioid.Pr
escriber","opiate")
clin_opiate=clin_opiate[clin_opiate$Opioid.Prescriber==1,]

top50 = tbl_df(clin_opiate) %>% top_n(100,opiate)
unique(top50$Specialty)

top20 = tbl_df(clin_opiate) %>% top_n(20,opiate)
top20 = top20[order(top20$opiate,decreasing = TRUE),]
top20[,12]=NULL
top20[,6:10]=NULL
top20[,2:4]=NULL
plot.new()
grid.table(top20)



# Top deathrate for states
clin_opiate = cbind(info.all.back[,1:11],info.all$Opioid.Prescriber,info.all$opiate)
colnames(clin_opiate) =
c("State","NPI","Gender.F","Gender.M","Credentials","doctor","nurse","other","PA","pharm","Specialty","Opioid.Pr
escriber","opiate")
clin_opiate=clin_opiate[clin_opiate$Opioid.Prescriber==1,]

top5state = info.all.back[info.all.back$deathrate %in% sort(unique(info.all.back$deathrate),decreasing
=TRUE)[1:5],]
unique(top5state$State)

# Rank state data
OD.st = OD.state


for (i in 1:nrow(OD.state)){
  currow = info.all.back[as.character(info.all.back$State)==as.character(OD.st$State[i]),]
  OD.st$medOpiate[i] = round(median(currow$opiate))
  OD.st$avgOpiate[i] = round(sum(currow$opiate)/ nrow(currow),2)
```

```
  # OD.st$deathrate[i] = as.numeric(gsub(",","",OD.st$Deaths[i],fixed=TRUE)) /
as.numeric(gsub(",","",OD.st$Population[i],fixed=TRUE))
  OD.st$deathper100k[i] = round(as.numeric(gsub(",","",OD.st$Deaths[i],fixed=TRUE)) /
(as.numeric(gsub(",","",OD.st$Population[i])) / 100000),2)
}

OD.st$rank.avgOpiate = rank(-OD.st$avgOpiate)
OD.st$rank.medOpiate = rank(-OD.st$medOpiate)
OD.st$rank.deaths = rank(-OD.st$deathper100k)

plot.new()
g1 = tableGrob(OD.st[order(OD.st$rank.avgOpiate)[1:5],],rows=NULL)
g2 = tableGrob(OD.st[order(OD.st$rank.medOpiate)[1:5],],rows=NULL)
g3 = tableGrob(OD.st[order(OD.st$rank.deaths)[1:5],],rows=NULL)

grid.arrange(g1,g2,g3, nrow=3, top="States Ranked by Avg/Med Opiates Prescribed(Upper/Mid) and
Deaths/100K(Lower)")
plot.new()
grid.table(OD.st[order(OD.st$rank.deaths)[1:5],])

OD.st[order(OD.st$rank.avgOpiate)[1:5],]
OD.st[order(OD.st$rank.deathrate)[1:5],]
```

```
#####
# Final Project
# Regression Analysis

clin_opiate = info.percent
clin_opiate = info.class
spec=c("Pain Management", "Interventional Pain Management", "Anesthesiology",
       "Physical Medicine and Rehabilitation Family Practice", "General Surgery", "Family Practice", "Internal
Medicine",
       "Rheumatology", "Physical Medicine and Rehabilitation", "Addiction Medicine", "General Practice",
"Neurology",
       "Emergency Medicine","Orthopedic Surgery", "Infectious Disease","Geriatric Medicine")
st = c("KY", "NH", "NM", "OH", "WV")

# clin_opiate = clin_opiate[info.all.back$Specialty %in% spec,]
#clin_opiate = clin_opiate[clin_opiate$State %in% st,]

#clin_opiate10 = clin_opiate[clin_opiate$opiate>50,]
#clin_opiate = clin_opiate[info.class$opiate<500,]

# Choose either Pain Management, Anesthesiology, Interventional Pain Management, or Physical Medicine and
Rehabilitation Family Practice

mydata = clin_opiate

set.seed(101) # reproduceable dataset

# Separate 80%/20% of data as training/testing
sample <- sample.int(n = nrow(mydata), size = floor(.75*nrow(mydata)), replace = F)
train.info.class <- mydata[sample, ]
test.info.class  <- mydata[-sample, ]

# Separate our data into predictors and responses
x.train = train.info.class[,2:73]
# x.train = train.info.class[,3:13]
# x.train$opiate=NULL
x.train$Opioid.Prescriber=NULL

y.train =train.info.class$deathper100k
# y.train =train.info.class$opiate
# y.train=as.numeric(gsub(",","",train.info.class$Deaths,fixed=TRUE))

x.test = test.info.class[,2:73]
# x.test = test.info.class[,3:13]
# x.test$opiate=NULL
x.test$Opioid.Prescriber=NULL
# x.test = cbind(test.info.class[,3:11],test.info.class[,13])

y.test =test.info.class$deathper100k
# y.test =test.info.class$opiate
# y.test=as.numeric(gsub(",","",test.info.class$Deaths,fixed=TRUE))

# library(PerformanceAnalytics)
# pairs(cbind(x.test,y.test),pch=".")

# Import libraries
library(glmnet)
library(ElemStatLearn)


# Linear Regression
lmfit=lm(y~. , data=data.frame(x=x.train,y=y.train))

# AIC & BIC Analysis
lmfitAICstep= step(lmfit , direction="both",trace=0, k=2)
lmfitAICfor = step(lm(y~1, data=data.frame(x=x.train,y=y.train)) , scope=list (upper=lmfit , lower=~1),
direction="forward",trace=0)

# BIC
lmfitBICstep= step(lmfit , direction="both",trace=0, k=log(ncol(x.train)))
lmfitBICfor = step(lm(y~1, data=data.frame(x=x.train,y=y.train)) , scope=list (upper=lmfit , lower=~1),
direction="forward",trace=0, k=log(ncol(x.train)))

# Ridge Regression
ridge.fit=cv.glmnet(y=y.train,x=as.matrix(x.train),alpha = 0, nfolds=5)
round(coef(ridge.fit, s = "lambda.min"),3)
ridge.fit$lambda.min

# Lasso Regression
```

```
lasso.fit = cv.glmnet(x=as.matrix(x.train),y=y.train, nfolds = 5)
round(coef(lasso.fit, s = "lambda.min"),3)
lasso.fit$lambda.min

# Predictions
pred.lm        = predict(lmfit,data.frame(x=x.test))
pred.AICstep   = predict(lmfitAICstep,data.frame(x=x.test))
pred.AICfor    = predict(lmfitAICfor,data.frame(x=x.test))
pred.BICstep   = predict(lmfitBICstep,data.frame(x=x.test))
pred.BICfor    = predict(lmfitBICfor,data.frame(x=x.test))
pred.ridge     = predict(ridge.fit,newx=as.matrix(x.test),s="lambda.min")
pred.lasso     = predict(lasso.fit,newx=as.matrix(x.test),s="lambda.min")

mselm          = mean((pred.lm - y.test)^2)
mseAICstep     = mean((pred.AICstep - y.test)^2)
mseAICfor      = mean((pred.AICfor - y.test)^2)
mseBICstep     = mean((pred.BICstep - y.test)^2)
mseBICfor      = mean((pred.BICfor - y.test)^2)
mseridge       = mean((pred.ridge - y.test)^2)
mselasso       = mean((pred.lasso - y.test)^2)

#
library(gridExtra)
library(grid)

mseresults= matrix(c(mselm,mseAICstep,mseAICfor,mseBICstep,mseBICfor,mseridge,mselasso),ncol=1,byrow=TRUE)
mseresults=round(mseresults,5)
colnames(mseresults) = c("MSE")
rownames(mseresults) = c("Linear Reg","AIC Step","AIC Forward","BIC Step","BIC Forward","Ridge Reg","Lasso Reg")
mseresults = as.table(mseresults)
plot.new()
grid.table(mseresults)


# rf = randomForest(x.train,y.train,ytest=y.test, xtest=x.test, importance = TRUE, ntree = 500)

# knn
library(kknn)

msetest=0
K=5
# pred = matrix(NA, K, length(x))

for ( k in 1:K ) {
  knn.fit = kknn(formula=y~., train=cbind(data.frame(y=y.train),x.train),
test=cbind(data.frame(y=y.test),x.test), k = k*10, kernel= "rectangular",distance=1)
  # pred[k,] = knn.fit$fitted.values
  msetest[k] = mean((knn.fit$fitted.values - y.test)^2)
}

mseresults2= matrix(msetest,ncol=1,byrow=TRUE)
mseresults2=round(mseresults2,5)
colnames(mseresults2) = c("MSE")
rownames(mseresults2) = c("k=10","k=20","k=30","k=40","k=50")
mseresults = as.table(mseresults2)
plot.new()
grid.table(mseresults2)
```

```
#####
# Final Project
# Cluster Analysis

#just need info.all
final_index <- length(info.all)-3
info.clustering <- info.all[,c(2:final_index)]
for (i in c(1:(final_index - 1))) {
  print(i)
  if (typeof(info.clustering[,i][2]) != "double") {
    info.clustering[,i] = as.numeric(info.clustering[,i])
    info.clustering[,i] <- sapply(info.clustering[, i], as.numeric)
    print("Done")
  }
}


info.clustering
spam.kmeans <- kmeans(info.clustering, centers = 5, nstart = 20, trace = TRUE)
spam.kmeans$centers
ggplot(info.clustering, aes(Population, deathrate)) +
  geom_point(col = c("blue", "red", "green", "orange", "purple")[spam.kmeans$cluster])

spam.kmeans$cluster
spam.kmeans$totss
sum(spam.kmeans$withinss)
vector <- rep(NA, 20)
for (i in c(1:20)) {
  spam.kmeans_sim <- kmeans(info.clustering, centers = i, nstart = 20, trace = TRUE)
  vector[i] = sum(spam.kmeans_sim$withinss)
}
scale(vector)
mse_kmeans = as.data.frame(cbind(as.numeric(c(1:20)), as.numeric(scale(vector))))
colnames(mse_kmeans) <- c("x", "y")
ggplot((mse_kmeans), aes_string("x", "y")) +
  geom_point() +
  scale_x_continuous(name="Number of clusters", limits=c(1, 20)) +
  ylab("MSE") +
  ggtitle("MSE vs Number of Clusters") +
  theme_bw()


final_index_clustring <- length(info.class)-3

info.clustering.class <- info.class
info.clustering.class$Population <- NULL
info.clustering.class$State <- NULL
info.clustering.class$Deaths <- NULL

for (i in c(1:length(colnames(info.clustering.class)))) {
  print(i)
  if (typeof(info.clustering.class[,i][2]) != "double") {
    info.clustering.class[,i] = as.numeric(info.clustering.class[,i])
    info.clustering.class[,i] <- sapply(info.clustering.class[, i], as.numeric)
    print("Done")
  }
}


vector.class <- rep(NA, 20)
for (i in c(1:20)) {
  spam.kmeans_sim <- kmeans(info.clustering.class, centers = i, nstart = 200, trace = TRUE)
  vector.class[i] = sum(spam.kmeans_sim$withinss)
}
mse_kmeans_class = as.data.frame(cbind(as.numeric(c(1:20)), as.numeric(scale(vector.class))))
colnames(mse_kmeans_class) <- c("x", "y")
ggplot((mse_kmeans_class), aes_string("x", "y")) +
  geom_point() +
  scale_x_continuous(name="Number of clusters", limits=c(1, 20)) +
  ylab("WSS") +
  ggtitle("WSS vs Number of Clusters") +
  theme_bw()

#pkgs <- c("factoextra",  "NbClust")
#install.packages(pkgs)
library(factoextra)
library(NbClust)
set.seed(101) # reproducible dataset
```

```
# Separate 80%/20% of data as training/testing
sample <- sample.int(n = nrow(info.clustering.class), size = floor(.50*nrow(info.clustering.class)), replace =
F)
subset.info.clustering <- info.clustering.class[sample, ]
nrow(subset.info.clustering)
plot_table<- as.data.frame(subset.info.clustering)
for (i in c(1:length(colnames(plot_table)))) {
  print(i)
  if (typeof(plot_table[,i][2]) != "double") {
    plot_table[,i] = as.numeric(plot_table[,i])
    plot_table[,i] <- sapply(plot_table[, i], as.numeric)
    print("Done")
  }
}
method <- fviz_nbclust(plot_table, kmeans, method = "wss",k.max = 20)
method_silhouette <- fviz_nbclust(plot_table, kmeans, method = "silhouette",k.max = 20)
fviz_cluster(kmeans(info.clustering.class, centers = 4, nstart = 25), data = info.clustering.class)

n_clust_silhouette <-method_silhouette$data
max_cluster_silhouette<- as.numeric(n_clust_silhouette$clusters[which.max(n_clust_silhouette$y)])

n_clust_wss <-method$data
max_cluster_wss<- as.numeric(n_clust_wss$clusters[which.max(n_clust_wss$y)])

method + geom_vline(xintercept = 3, linetype = 2) +labs(subtitle = "Elbow method")
method_silhouette + geom_vline(xintercept = 2, linetype = 2) +labs(subtitle = "Silhouette")

print(max_cluster_silhouette)
print(max_cluster_wss)

library(cluster)
gap_stat <- clusGap(plot_table, FUN = kmeans, nstart = 25, K.max=20, B=20)
print(gap_stat, method="firstmax")
fviz_gap_stat(gap_stat)
```

```
#####*************************************************************************************************
# Unsupervised Learning- Principal Component Analysis
#*********************************************************
# Final Project
# Data Setup
# # 1. IMPORT DATA
setwd("G:/My Drive/PhD/Class/Statistical Machine Learning/Final Project")
setwd("us-opiate-prescriptions")
opioids = read.csv("opioids.csv")
info = read.csv("prescriber-info.csv")
overdose = read.csv("overdoses.csv")
class = read.csv("drugclass.csv")
cred = read.csv("credentials.csv")
spec = read.csv("specialty.csv")
colnames(class)[1] = "Drug.Name"

setwd("..")
setwd("Source Codes")
getwd()
source("SPM_Panel.R")
source("PCAplots.R")
source("FactorPlots.R")
source ("TestSet.R")
source ("pc.glm.R")
source("ROC.R")

setwd("..")
getwd()
class = read.csv("drugclass.csv")
colnames(class)[1] = "Drug.Name"

# Libraries
library(hashmap)
library(plyr)
library(plotly)

# add drug class to the info matrix
classunique=unique(class$Drug.Class)
info.drug = colnames(info[,6:255])
classmask = data.frame(row.names=classunique)

for (c in 1 : length(classunique)) {
  r = info[1,6:255]
  rownames(r) = classunique[c]
  for (k in 1:length(info.drug)) {
    if (info.drug[k] == class$Drug.Name[k] && class$Drug.Class[k] == classunique[c]){
      r[k] = 1
    } else {
      r[k] = 0
    }
  }
  classmask = rbind(classmask,r)
}

# Apply mask
info.all = cbind(info,data.frame(t(as.matrix(classmask) %*% as.matrix(t(info[,6:255])))))
info.class = cbind(info.all[,1:5], info.all[,256:318])

# Clean up credentials - add in rows
credunique=unique(cred$type)
credh = hashmap(cred$credential,cred$type)
credmat = as.data.frame(matrix(0,ncol=length(levels(credunique)),nrow=nrow(info.all)))
colnames(credmat) = levels(credunique)
for (k in 1:nrow(info.all)) {
  credmat[k,credh[[info.all$Credentials[k]]]]=1
}

info.all = cbind(info.all[,1:4],credmat,info.all[,5:length(info.all)])
info.class = cbind(info.class[,1:4],credmat,info.class[,5:length(info.class)])

# Clean up Specialty
specmat = as.data.frame(matrix(0,ncol=ncol(spec)-1,nrow=nrow(info.all)))
colnames(specmat) = colnames(spec)[2:ncol(spec)]

for (i in 1:(ncol(spec)-1)) {
  spech = hashmap(spec$Specialty,spec[,i+1])
  for (k in 1:nrow(info.all)) {
    specmat[k,i]=spech[[info.all$Specialty[k]]]
  }
```

```
}

info.all = cbind(info.all[,1:10],specmat,info.all[,11:length(info.all)])
info.class = cbind(info.class[,1:10],specmat,info.class[,11:length(info.class)])

# Add in death rates
OD.state = overdose[,2:4]
colnames(OD.state) <- c("Population", "Deaths", "State")

info.all <- merge(info.all, OD.state)
info.all$deathrate <- as.integer(info.all$Deaths) / as.integer(info.all$Population)
info.class <- merge(info.class, OD.state)
info.class$deathrate <- as.integer(info.class$Deaths) / as.integer(info.class$Population)

# Convert M to 1 and F to 2
info.all$Gender = revalue(info.all$Gender,c("M" = "1", "F" = "2"))
info.class$Gender = revalue(info.class$Gender,c("M" = "1", "F" = "2"))

info.all.back = info.all

# If you want to work with JUST numbers, use these dataframes
# info.all has ALL drug data and drug classification data
# info.class has just drug classification data

info.all$Specialty = NULL
info.all$Credentials = NULL
info.class$Specialty = NULL
info.class$Credentials = NULL

# ********************************************************************************************
# Unsupervised Learning_ Prinipal Component Analysis
library("FactoMineR")
info.practice.all <- info.all[,12:323]
for (i in c(1:length(colnames(info.practice.all)))) {
  if (typeof(info.practice.all[,i][2]) != "double") {
    info.practice.all[,i] = as.numeric(info.practice.all[,i])
    info.practice.all[,i] = sapply(info.practice.all[,i], as.numeric)
  }
}
Opioid.PC = prcomp(info.practice.all, cor = T)
summary(Opioid.pc)
biplot(Opioid.PC)

library("factoextra")
get_pca(Opioid.pc, element = c("var", "ind"))
fviz_eig(Opioid.pc)

screeplot(Opioid.pc, main = "Variance for PC of Metrics")
# plot(loadingsplot(Opioid.pc))
cum_prop <- cumsum(Opioid.pc$sdev^2 / sum(Opioid.pc$sdev^2))
last_index = min(which(cum_prop > 0.90))
Opioid.pc = princomp(info.practice.class, cor = T)
loadingsplot(Opioid.pc)
sort(Opioid.pc$loadings[,1])

# Different way of Analayzing the data with k-means clustering
Opioid.pca = PCA(info.practice.all,  graph = F)
print(Opioid.pca)
# Visualization
fviz_pca_biplot(Opioid.pca)

# Eigenvalues
library("factoextra")
eig.val <- get_eigenvalue(Opioid.pca)
eig.val
fviz_eig(Opioid.pca, addlabels = TRUE, ylim = c(0, 50))

var <- get_pca_var(Opioid.pca)
var

# Create a grouping variable using kmeans
# Create 3 groups of variables (centers = 3)
set.seed(123)
res.km <- kmeans(var$coord, centers = 3, nstart = 25)
grp <- as.factor(res.km$cluster)
# Color variables by groups
fviz_pca_var(Opioid.pca, col.var = grp,
             palette = c("#0073C2FF", "#EFC000FF", "#868686FF","red", "green"),
             legend.title = "Cluster")
```

```
res.desc <- dimdesc(Opioid.pca, axes = c(1,2), proba = 0.05)
# Description of dimension 1
res.desc$Dim.1

#### ***************************************************************************************
# Create the dataframe of deaths and opiates
Opideaths<-data.frame(info.all$Deaths, info.all$opiate)
#principal components on Opideaths
info.practice <- Opideaths
for (i in c(1:length(colnames(info.practice)))) {
  if (typeof(info.practice[,i][2]) != "double") {
    info.practice[,i] = as.numeric(info.practice[,i])
    info.practice[,i] = sapply(info.practice[,i], as.numeric)
  }
}
Opideaths.pc = prcomp(info.practice, cor = T)
par(mfrow=c(1,1))
biplot(Opideaths.pc, xlim  = c(-0.1,0.3), ylim  = c(-0.05, 0.05))
cumplot(Opideaths.pc)

# ggplot
plot <- data.frame(PC1=Opideaths.pc$x[,1],PC2=Opideaths.pc$x[,2])
ggplot(plot, aes(x = Opideaths.pc$x[,1], y = Opideaths.pc$x[,2]))


#### ***************************************************************************************
# PCA on info.class
library(dplyr)
info.class1<-select(info.class, -c(State:NPI,Population:Deaths))

info.practice.class <- info.class1[,12:72]
for (i in c(1:length(colnames(info.practice.class)))) {
  if (typeof(info.practice.class[,i][2]) != "double") {
    info.practice.class[,i] = as.numeric(info.practice.class[,i])
    info.practice.class[,i] = sapply(info.practice.class[,i], as.numeric)
  }
}

Opioid.pc = prcomp(info.practice.class, cor = T)
# loadingsplot(Opioid.pc)
# sort(Opioid.pc$loadings[,1])

# head(info.full[,3:68])
par(mfrow=c(1,1))
biplot(Opioid.pc)

cumplot(Opioid.pc)
# loadingsplot(Opioid.pc)
cum_prop <- cumsum(Opioid.pc$sdev^2 / sum(Opioid.pc$sdev^2))
last_index = min(which(cum_prop > 0.90))

###
library(dplyr)
# Creating a binary values for opiate greater than 10 and less than 10
Opiatecolor<-ifelse(info.class$opiate>10,1,0)
plot <- data.frame(PC1=Opioid.pc$x[,1],PC2=Opioid.pc$x[,2], label=as.factor(Opiatecolor))

# colnames(plot) <- c("PC1", "PC2")
ggplot(plot, aes(PC1,PC2, color=label))+geom_point()


# Creating a binary values for opiate greater than 50 and less than 50
Opiatecolor<-ifelse(info.class$opiate>50,1,0)
plot <- data.frame(PC1=Opioid.pc$x[,1],PC2=Opioid.pc$x[,2], label=as.factor(Opiatecolor))

# colnames(plot) <- c("PC1", "PC2")
ggplot(plot, aes(PC1,PC2, color=label))+geom_point()

# Creating a binary values for opiate greater than 500 and less than 500
Opiatecolor<-ifelse(info.class$opiate>500,1,0)
plot <- data.frame(PC1=Opioid.pc$x[,1],PC2=Opioid.pc$x[,2], label=as.factor(Opiatecolor))

# colnames(plot) <- c("PC1", "PC2")
ggplot(plot, aes(PC1,PC2, color=label))+geom_point()

# Creating a binary values for opiate greater than 1000 and less than 1000
Opiatecolor<-ifelse(info.class$opiate>1000,1,0)
```

```
plot <- data.frame(PC1=Opioid.pc$x[,1],PC2=Opioid.pc$x[,2], label=as.factor(Opiatecolor))

# colnames(plot) <- c("PC1", "PC2")
ggplot(plot, aes(PC1,PC2, color=label))+geom_point()
#*********************************************************************************************
```