USA Spending Data Analytics for Business Owners

By Shao-Hung Hsiao (913293454), Jared Yu (914640019), Haoran Zhang (913074327)

Mar 22, 2019

Abstract

This project presents an investigation of the U.S. government spending in recent years on business companies. The question that we are interested in finding more information about is, ”How do different variables representing various companies and the U.S. government affect the federal spending levels.” More precisely, we were looking to model the data in such a way that an interested business would be able to compare the characteristics of their company and potentially make business decisions based on insights gathered from our data analysis. This project mainly focus on four such variables: state, category, sector, and government department. Various analytical methods have been utilized to address such questions including: time series analysis, text mining , machine learning, etc. In our examination of the dataset we were unable to find a conclusive decision based on our research methodology and original beliefs regarding the model parameters. The report below outlines first an introduction to our goals, then a methods section which explains the background of our approach to different problems, followed by the results of our analysis. This projects then concludes with our main points and any further discussion about areas to improve upon given additional time.

Introduction

The dataset consists of a list of transactions paid for by the federal government, it is mainly composed of contract work to individual private companies. Looking at the available data, we are interested in providing insight to businesses that are interested in receiving this type of federal funding from the U.S. government. Our goal then is to build different models that help potential business owners determine how they can start their business, or help existing business owners to expand. Specifically, we want to help them discover their prospects by state, sector, and depending on which business categories

they are associated with. Better understanding this information will give business owners insight into where they can possibly receive the most money from government departments for local investment.

Methods / Data

The original dataset includes the data of up to 1973 based on fiscal year. However, data that is too far away will not be useful for our prediction or for potential business owners. We only use the data in this decade (2011-2018) and to improve accuracy, 2019 data is not used because it is not completed yet. Additionally, we wish to use annual data. The number of data subsetted are roughly evenly distributed based on the fiscal year:

2011	2012	2013	2014
7850349	6399715	5946744	6343613
2015	2016	2017	2018
7698699	8446348	8783688	11687701

From the official documentation(Interface Definition Documentation, IDD), total_obligation is the summation of federal_action_obligation that belongs to the same award_id, so we decide to use total_obligation as the response variable because determining the final money received is more meaningful than to predict the money received for each record. It also explains why there are many transactions with repeated total_obligation in the previous homework that we were not able to find out the reason, like the 330 repeat records for sandia national lab. Also from IDD, NAICS_code, business_category, awarding_agency_toptier_name, and recipient_location_state_code meet our interest to approach the big question. Therefore, we would like to mainly focus on those variables. The original sqlite database does not contain the information we need when we started. Therefore, it is not used in this project.

For total_obligation, there are some negative values. According to the usaspending website, “there may be an initial award that was reported in the past but the current data only shows a de-obligation transaction, thus resulting in a seemingly net negative award.” Without seeing the whole picture, the negative values are thereby removed along with NA’s, which accounts for 2.13% of the total data. The total_obligation is inflation adjusted based on their first fiscal years when the award is made. It would represent how much money the government is willing to spend more accurately.

Time Series Forecasting

The dataset contains variables which are related to time and therefore allow the dataset to be expressed by certain time order. This would allow the data to be interpreted using time series analysis. It has been mentioned that the time frame which we are interested in is from 2011-2018. Also, we would like to analyze the individual trend for each state within this time period. Using this portion of data, we are interested in understanding the total spending from the government. The general model for time series model is as follows,
$Y_t = s_t + m_t + X_t$

where Y_t are the observations, m_t is the seasonal component, is the trend component, and X_t is the rough component. We will be utilizing the yearly component which reduces our sample data to only 8 observations. For this reason, it would be difficult to calculate any realistic seasonality and so the seasonal component will not be included in the analysis. There are many possible methods for identifying a trend, these include: Loess, polynomial, and two-sided moving average. In general, Loess is the preferred method. Loess stands for locally weighted moving average, but rather than fitting an individual Loess to each of the 50 states, we will use another method instead.

The goal is to have a rough component which can be understood as stationary. Stationary can be understood roughly as for when a series has a constant mean and variance through the data. This is an important trait if we wish to do prediction. We will not go too deep into the math or the reasoning, but it is like the normality of the error term in linear regression. Instead of finding a unique trend for each of the 50 states, we will instead use first-order differencing. The model can be understood as follows,
$X_t = \ \bigtriangledown \ Y_t = Y_t - Y_{t-1}$

where the rough at the time t is identified as being the difference of observation t with observation $t - 1$. This differencing of the series will lead to the rough being shorter by one observation. It is possible to do second or even third order differencing, but neither case is common. Therefore, we will use this first order differencing as a first-step in detrending to find a stationary rough component. To further model this rough component, we will utilize $ARMA(p, q)$ to determine how to next best fit the rough. Since we have already taken the difference, the data is currently modeled as $ARIMA(p, d = 1, q)$. We will further determine the necessary values for p and q by fitting a series of 25 possible models of $0, \cdots, 4$ for each combination of p and q leading to a distinct *AICc* criterion each time we fit a model. The lowest value will lead to the decision for which type of $ARIMA(p, 1, q)$ model we decide to fit the rough with. The $AR(p)$ and $MA(q)$ components can be understood as

autoregressive and moving averages. Without getting into too much detail, the two models have the following appearance,

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t$$

$$X_t = \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p} + \varepsilon_t$$

where the first model is an $AR(p)$ model and the second is an $MA(q)$ model. The first model shows that the current observation can be predicted based on previous observations.The second observation says that we can understand the current observation based on the white noise $\varepsilon_t \sim WN(0, \sigma^2)$ from previous observations. They can be combined into the following,

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p}$$

which is the $ARMA(p, q)$ model. Without going too deep, these can be seen visually using autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. These plots show how much lag exists at lag for the previous observation $t - j$.

The goal of our use of time series methods is not to only model the rough component, but to develop predictions on the future annual government spending by each state. To do this, we will utilize a more convenient function in R called sarima.for(), which will perform forecasting of future observations given an $ARIMA(p, d, q)$ model. We will use the function to estimate future spending based on the model that we have specified for each state. The number of years we will forecast is 5 years into the future.

Geo Plots to Understand Sectors Across the Country

We are analyzing different state data to gain a better understanding of the different possible business opportunities for companies who are interested in new investments. It would be beneficial then if these companies could gain a geographic view of the business landscape. We will utilize the plotly package to perform geo plotting of different business sectors based upon their North American Industry Classification System (NAICS) code. The NAICS code has been modeled with the more general Standard Industrial Classifications (SIC) division. Using Geo Plotting on the sector information can provide us with valuable instantaneous insight into where there is a strong demand for a given business' sector. For example, if a company in the services sector is interested in finding out where states have a large proportion of their funding coming from the services sector, then they can simply look at the geographic plot of the country with colored graphics that can quickly tell the answer.

General Economic Data

In addition to providing information from within the dataset about the possible prospects of starting investments in a certain state, it would be good to have other general economic information to help understand the business environment. Using economic data from the API of the Federal Reserve of St. Louis, we can understand macroeconomic factors such as GDP, unemployment, and other variables which are of use. Visualization of these data can provide some extra analysis which can help to better understand the current business conditions.

We will analyze several of these variables in the broad macroeconomic sense so that businesses who are uncertain about their decision can be more confident in whether they decide to act on a potential investment prospect. This will primarily be done using visualizations of economic data and the analysis of which can be done using macroeconomic theory.

Text Mining

We have two variables which consist of a string of words which are difficult to interpret when there is such a vast amount of data. To make sense of these descriptions business_categories and awarding_toptier_agency_name, we will utilize the text mining methods which were learned in a previous class assignment. We will filter the dataset annually and then again by each state. This will give us a better idea of the trends over time for each state. If a business starts to feel that a specific state is a good idea for their investment, we can look next to the most frequent types of descriptions that relate to these states by year.

The method utilizes what are called weighted character vectors. Each of the two columns mentioned consists of a long character string with different information. To understand a single row is a simple task and does not require work, however, to interpret a large amount of this information requires some technique. We will take each string as split them into separate words, removing spaces. Additionally, we will make use of ‘stem words’ from Natural Language Processing to cut down words to their basic components. For instance, the word business can be shortened to ‘busi-‘, so that words like business and businesses can be grouped together as a single stem word. Other techniques include removing common ‘stop words’ such as ‘if’ or ‘and’ so that they need not be weighted for their importance. In addition to having long strings of stem words, we can weight the importance of these stem words by relating them to their value in terms of total_obligation. For example, if a row is broken down into six stem words, we can divide the total

obligation for this observation by six and attach that value to each of the stem words. Then when done doing this task for a certain state on a specific year we can sum each of them stem words with their total values to determine which stem words are most 'valuable.' Looking at these proportions can give us an idea of what sort of business categories or awarding agency descriptions are common for a given state on a specific year. This information can give us insight into what sort of business conditions a company would like to know before they proceed with investing in a new area.

Machine Learning

The dataset for machine learning is cleaned in a special way. Since the dataset is 65GB uncompressed, it's too costly to use the entire dataset for cleaning purposes. Therefore, bash scripts are ran on cluster in order to extract the minimum information needed for cleaning each column, columns are combined and shuffled to form the dataset. Data loss is minimized by using and combining the non missing information belong to the same award_id.

All the independent variables are categorical variables, some of them has many levels. We reduce the levels in the following way: The levels(including NA since we do not want to lose much data) are ranked based on their frequencies, when the cumulative distribution reaches 95%, all the levels afterwards are generalized to one category since the number of observations of those levels would be too small for training and testing purposes. Ordinal encoding is used initially, results in 29848 bins(total combinations of those levels) in total. One Hot encoding is used later when actually train and test the data.

The first two digits of the naics_code is used to represent the sector that a business belongs to according to NAICS website. Unfortunately half of the data is missing, and this is the primary reason that we do not want to simply remove NA's in our dataset. Business_category is a very unorganized column because this is self-reported by the business at sam.gov, according to a community post on the usaspending website. We choose the first category appeared in the this column for the following reason: when self reporting the business type, the first word that being reported should best represent the type of a business.

Regression

- Linear Regression

Find the linear function
$y(x) = w^T x + \epsilon$

with w as the weight vector and x as the input vector. And ϵ is the residual error we trying minimized with Ordinary Least Squares
$w = (X^T X)^{-1} X^T Y$

- Gradient Boosting Tree

  It is a ensemble of multiple decision trees. The basic idea is to build decision tree to predict the loss of the previous model's prediction. And continue to the maximum number of iteration.

Classification

- Logistic Regression

  Similar to Linear regression, but instead of a polynomial function, we use the sigmoid function.
  $sigmoid(z) = 1/(1 + e^{-z})$

  Therefore, the logistic regression function would be
  $g(x, w) = 1/(1 + e^{-w^T x})$

  with w as the weight vector and x as the input vector. And we are looking for the decision boundary to separate the data into two classes.

  However, unlike the Ordinary Least Squares we have for Linear Regression, we formulate it as a Maximum Likelihood Estimation problem and used stochastic gradient descent to update the weight / minimize the error.

  We used one-vs-rest method for our model, which is basically means that we split the problem into 10 binary classification problems (since we have 10 classes).

- Support Vector Machine

  The basic idea of a Support Vector Machine for classification is to find the hyperplane in the n-dimensional that can use to classify two classes.

  Same as the Logistic Regression, the idea of stochastic gradient descent, and one-vs-rest method were used here as well.

- K-means Clustering

  As what we learned from class, it randomly select n center point and update the location of those center points based on the nearby data point until the cluster became stable. And data points are classify into different classes based on the closest center point.

  For this model we used for this project, the Elkan's algorithm to have better performance.

Evaluation Method

- 5-fold Cross-validation Accuracy with 95% confidence interval

  Data set were divided into 5 subsets evenly, then trained 5 times with different combination of training and testing sets.

  Accuracy = # *of correct prediction* / # *of prediction*

  Logistic Regression and Support Vector Machine both used this method to evaluate their performance.

- Mutual Information based scores

  Mutual Information ignore the permutation of class label to calculate the percentage of agreement between the true label and the clustered label.

  K-means was evaluated with this method.

First, we thought it would be a good idea to model this problem as a regression problem. Where we use features like state, funding agency, business category, and NAICS to predict the corresponding total obligation. And we decided to try with two different model, Linear regression and Xgboost.

However, the results were disappointed. Our Linear Regression model ended with the coefficient of determination R^2 of 0.0005, which is very close to result of always predicts the expected value of y, disregarding the input features. And xgboost model, which took 5 hours + to train, ended with a very similar mean square error we get at the beginning of the training.

After some discussion, we decided to model the problem into a classification problem. Mainly because that we are interested to know what how much money we can get from

the government for a company in x industry at y state. Since the amount of funding could be affect by many factors, predicting a range would be sufficient.

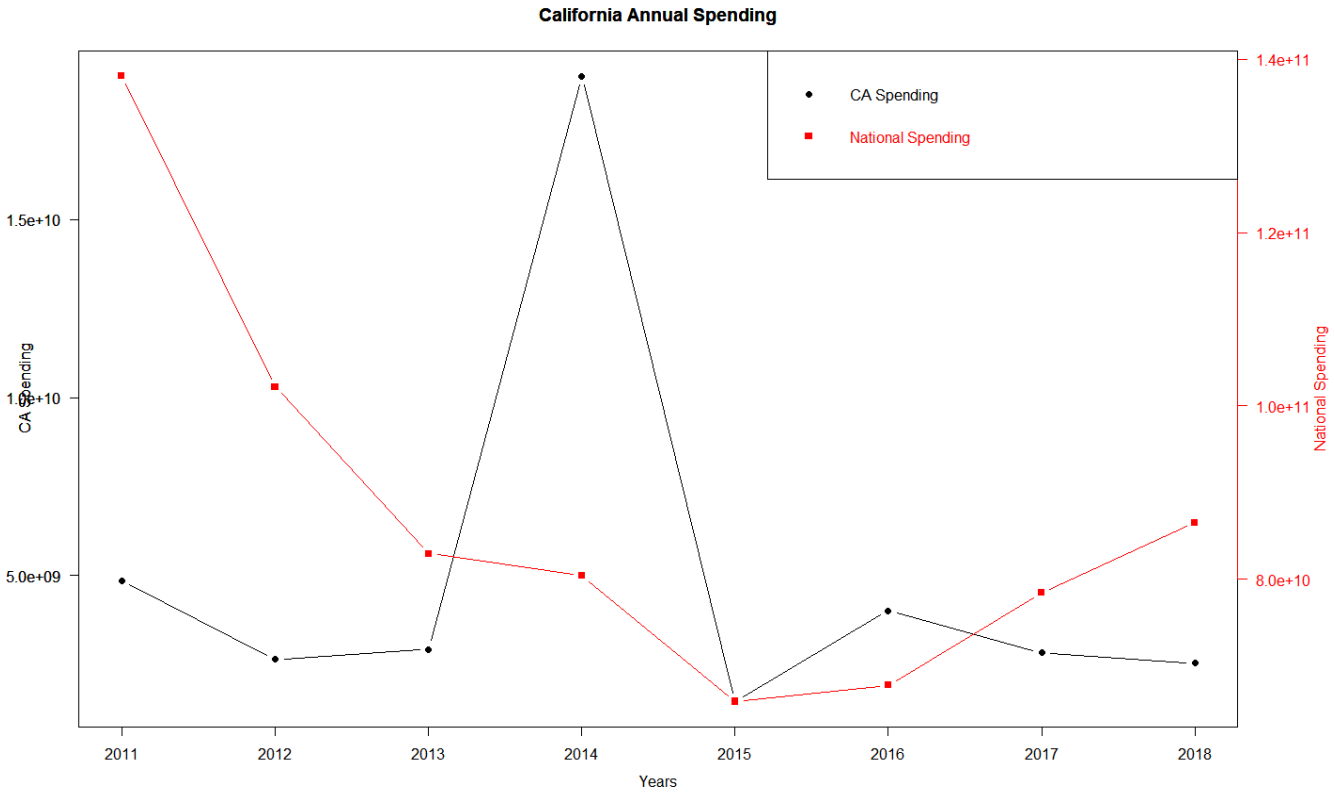
We also decided to remove state as one of the feature. First, because since the problem we are try to solve is specific to a particular state. There will be no reason for our the model on this particular state to trained with the data from other states. Second, it can reduce the amount of training time significantly.

Result

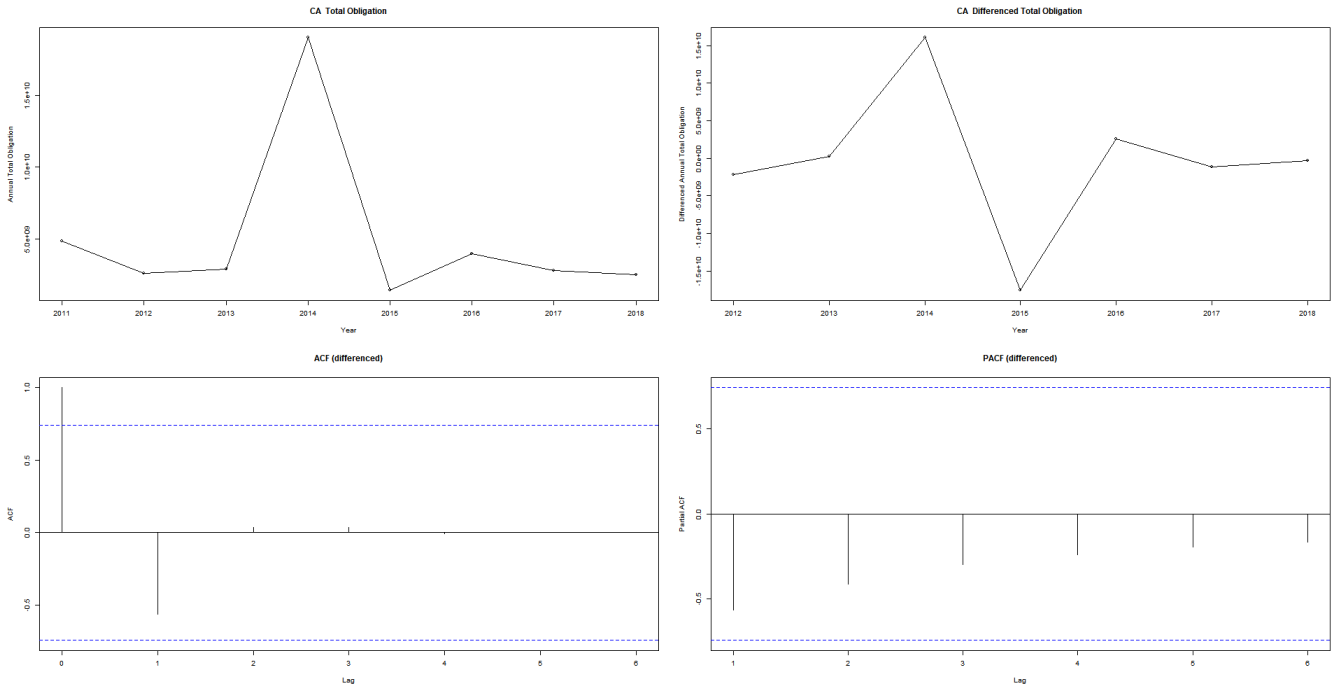
Below we will discuss the results from the different methods that we have discussed earlier. These results are output from our analysis using computational software such as R and Python. We will thoroughly describe what we had found and our conclusions based on our investigation into the data using our methodologies previously mentioned.

Forecasting with Time Series Data

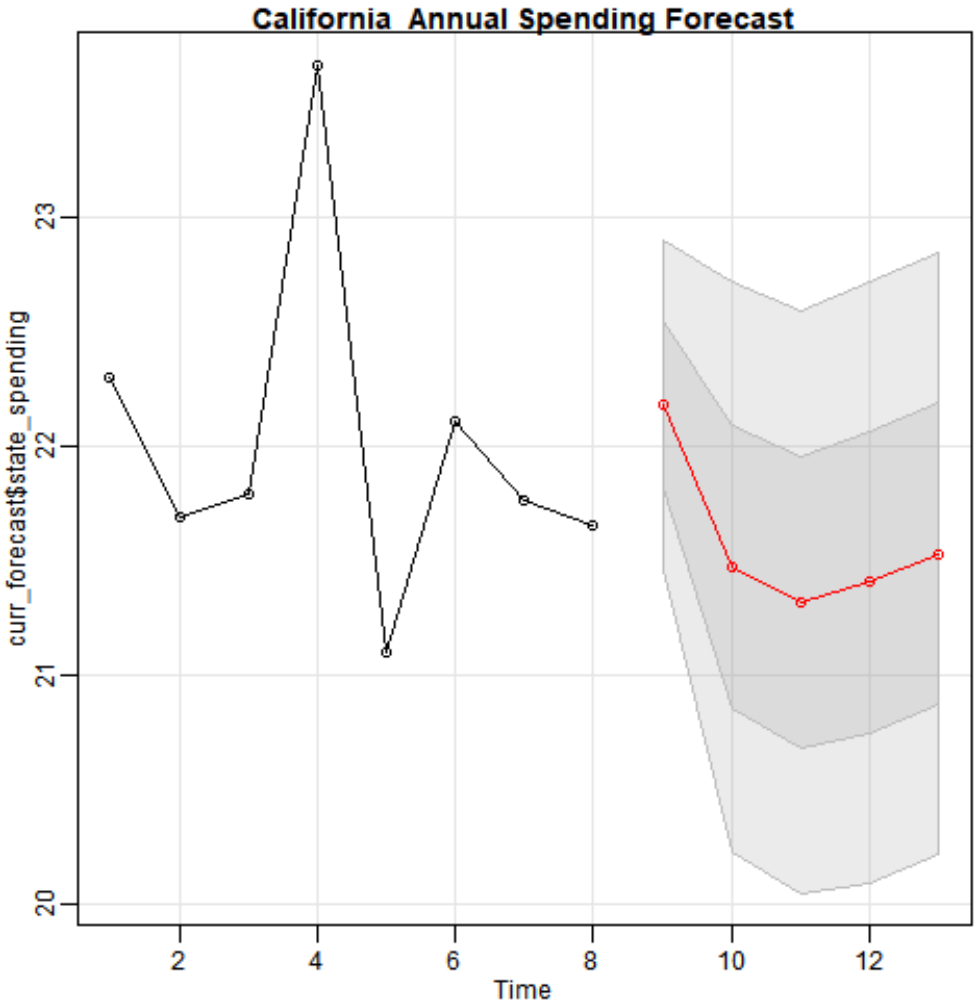
During the process of doing time series modeling, we are interested in finding out some information regarding the future trend of government spending. We have explained earlier that we would utilize forecasting with $ARIMA(p, 1, q)$ for each of the different states. The result of this had shown that many of the states exhibit a negative trend. A good start before we analyze the forecast to first examine the raw observations themselves. We can look at all 50 states, but that is difficult to understand. Instead, we will take the example of California, as it is the state which consistently has some of the largest GDP and spending throughout the country. Below we can see the annual spending for California, plotted alongside the trend of the annual spending for the entire country.



We see that California experienced a large spike in government spending between 2013 and 2015, while the rest of the country was on a downwards trajectory. This data is quite erratic and not smooth in terms of time series analysis. There is a combination of large spikes and strong trends, making it difficult to model in its current condition. To develop the more stationary rough component, we will next analyze the process of first order differencing.



The next plot shows both the raw observations in the top left of California's annual spending with the result of taking the first order difference on the right. The ACF and PACF plots of the first order differenced data shows that all the tails remain within the confidence intervals (except for the first tail at 1 for the ACF, but this is always the case and is meaningless to interpret). The plots show that the first order differenced data appears to pass certain tests for stationarity, but we will go further and do some modeling on the differenced data.

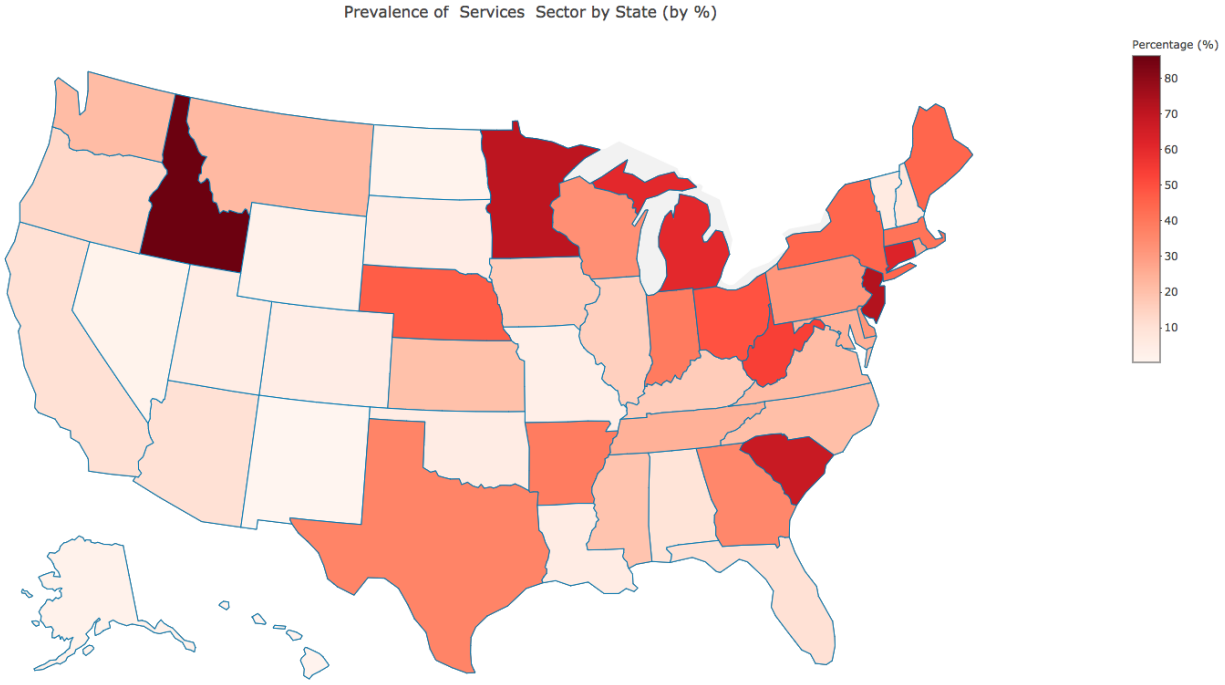


This plot does not show, but the model that was fitted based on the *AICc* criterion is $ARIMA(4,1,2)$. The above plot shows the forecasting of the next 5 years based upon that model. The y-axis is not labeled ideally, but this is a result of the function from the R package not allowing for this option. The numbers are also quite small, because the data has been transformed with the natural logarithm before doing forecasting. The reason is that there was a bug which seemed to occur in R when the computer uses matrix multiplication of a Hessian matrix, which seems to break down when numbers such as California's government spending are used. The natural logarithm breaks these numbers into smaller sizes which can be used to forecast. The interpretation should not be too

different and here it shows that there is a negative outlook. This is the majority interpretation for most of the states in the upcoming few years. Such information is useful for a prospective company because if they had believed they should go into a certain state to invest in their local economy, it could be a mistake. Looking at the current forecast of government spending, it may be not as likely that they will receive funding for their projects due to a slowing global economy.

Sector Analysis with Geo Plotting

With so much information available about spending from the government, we are interested in knowing what types of spending is going on. More specifically, for a company who is interested in expanding, what sort of areas are showing a strong presence of that sector.



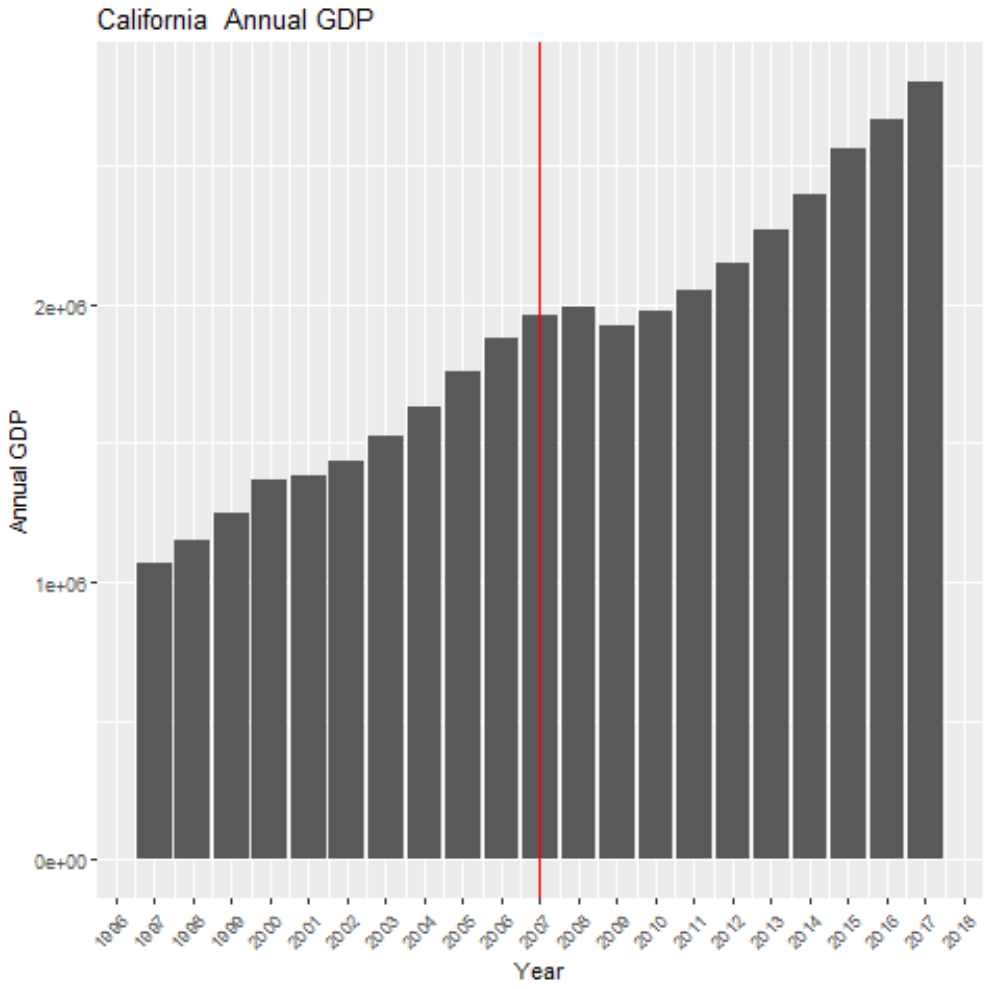
Above we can see the distribution of spending related to the services sector. In other words, if a company belongs to the services sector from the NAICS code, they may be interested in finding states which have a large percentage of their spending as coming from this sector. For instance, we can see that the east coast has a larger prevalence of this sort of funding going on and so a company would be interested in going there to possibly gain government contracts to do work.

This pattern could be repeated for any number of different sectors such as Agriculture, Mining, etc. Geo plotting is beneficial and unique in that we can instantly get a snapshot

of the geographic outlook for this sector at a given time. For instance, this data could be narrowed down to yearly values, to give us the projection of the trend over time.

Macroeconomic Analysis

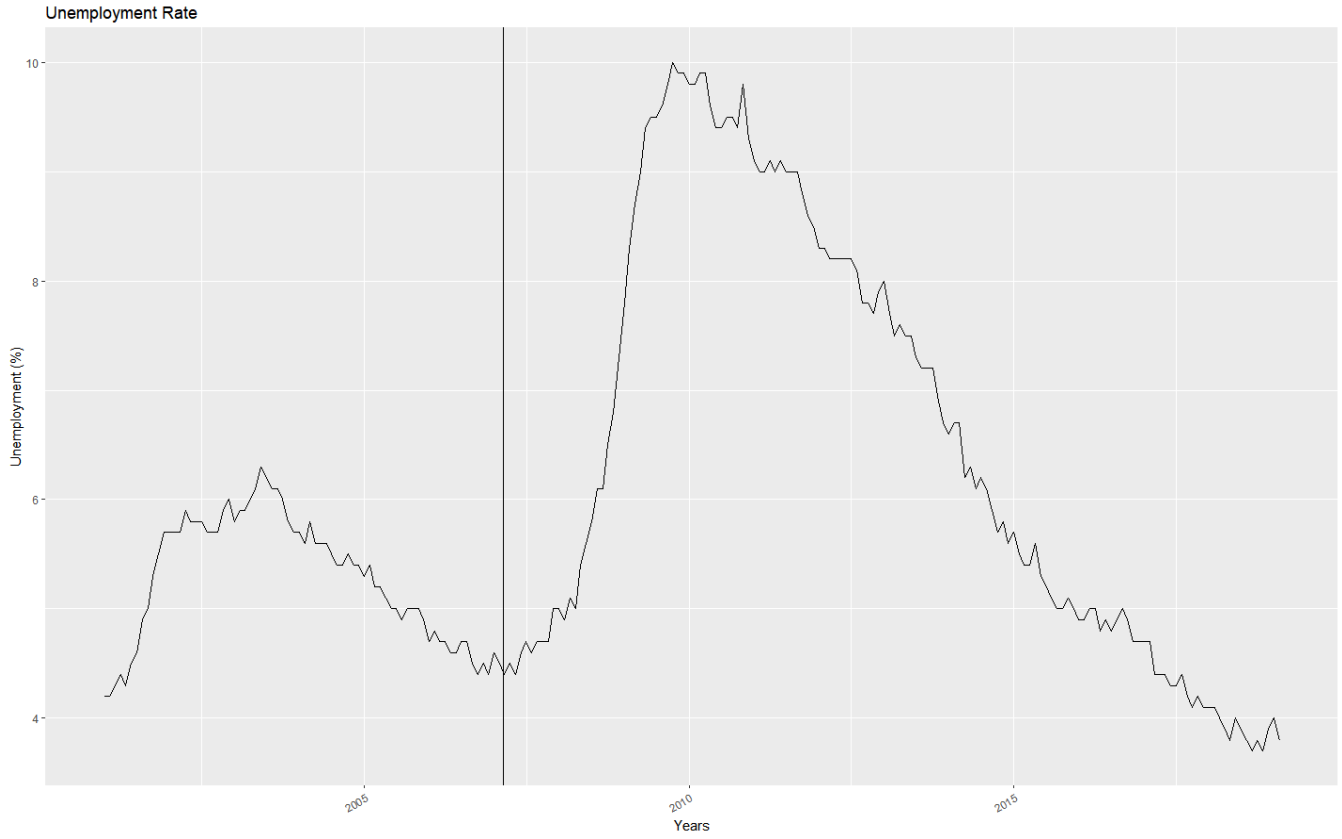
Much of the information we are currently discussing is in regard to possibly understanding an overall economic trend that may be worth taking advantage of. Being able to jump on a trend when it starts can be the breaking moment for when a company makes it into the future or struggles and gets left behind like so many others. We have investigated some macroeconomic data freely available through the Federal Reserve website to understand some gauges of the economy using other government measuring instruments.



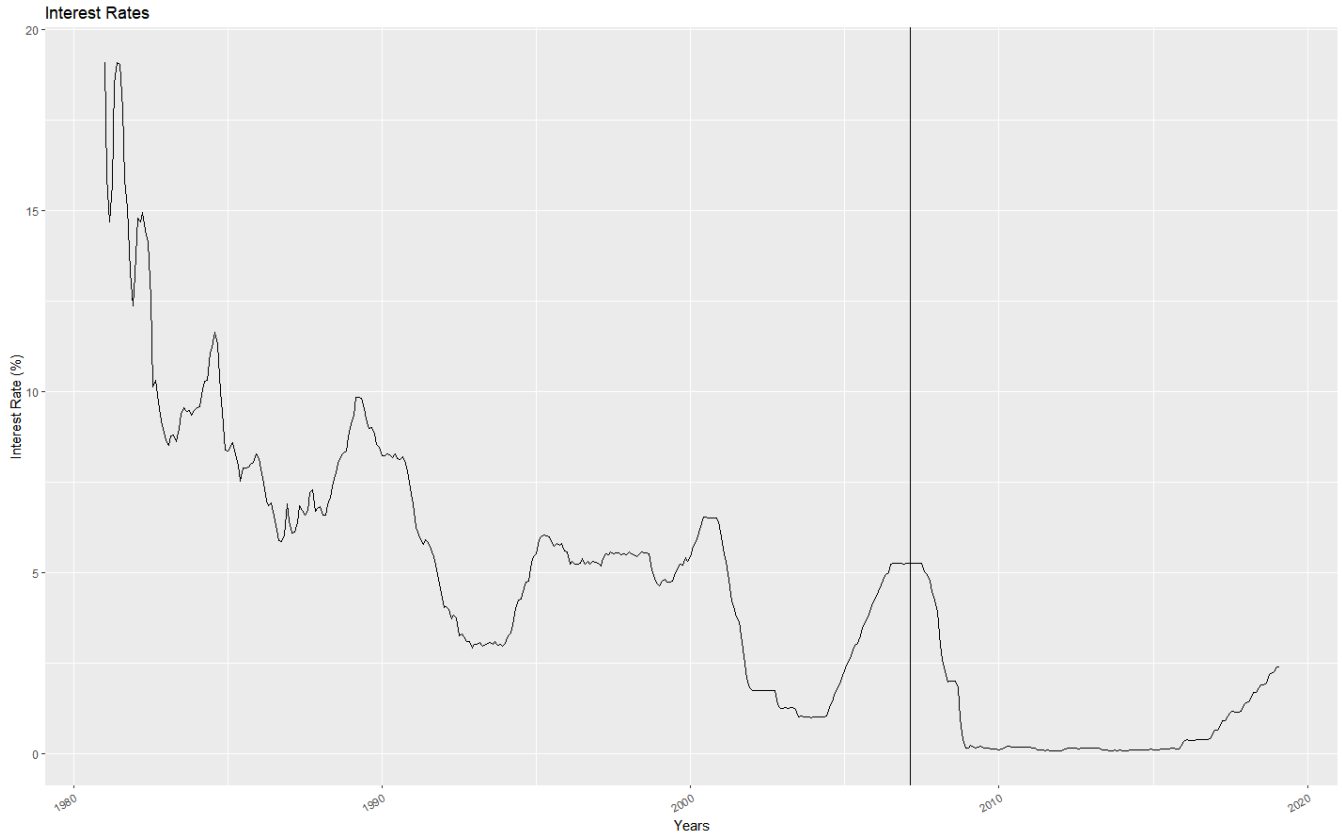
Above we have the annual Gross Domestic Product (GDP) of California since 1996. This information tells us about the goods that are produced by the economy of California and how the prices of products are increasing over time. It can be thought of one of the measures of inflation such as Consumer Price Index (CPI). The red line shows 2007, the year of the Global Financial Crisis. It is noticeable that shortly after the GDP of the state takes a dip before resuming an upwards trend. This is a common pattern for almost every

state during this period. It shows that the economy previously had a projected slope which it deviated from for some time before resuming an upwards trend.

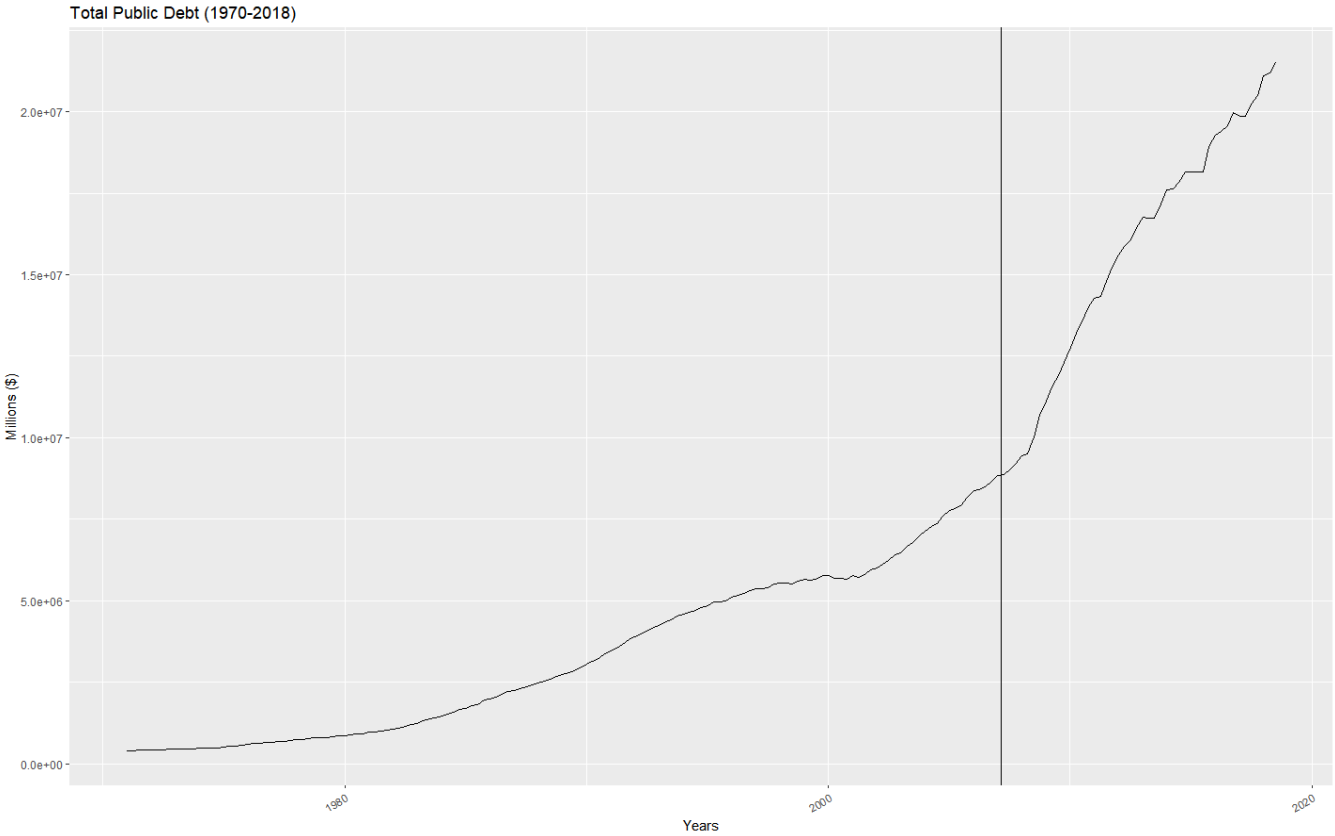
This sort of information may be interesting for a company that is trying to find out more information before risking more money on investments which may or may not have positive returns. If it were just after 2007, it may not have been certain at the time whether the economy had yet begun a real path of recovery. Such data can be thought of as information which a business would like to know and make use of to assist them in making a decision or not making a mistake.



The next plot shows the Unemployment Rate of the country from the year 2000 to the present. Again, a vertical line is drawn showing the time of the Global Financial Crisis. Afterwards, unemployment rose sharply up to the highest levels in years. It would have been a wise choice to perhaps instead not invest during this time, as people had no job and therefore no income to spend money for the company’s products.



The next plot shows interest rates for the country. Again, there is a line showing the time of the Global Financial Crisis and how interest rates plummeted to near zero. This plot would be important for a business so that they could understand that the low interest rates from Quantitative Easing on the part of the Central Bank are making for a more optimum business environment. With low interest rates, it is cheaper for a business to borrow money from the bank and so their future investments could have greater returns with lower risk.



This last plot of macroeconomic data shows the total public debt of the country. Since the Global Financial Crisis, the public debt has risen much faster at a steeper curve. The annual debt has surpassed the annual GDP of the country. Such information would be important for a company to know if they were interested in seeking government funds for contract work. This information may indicate that it is not right now a good time to seek funds from the government if they are suffering from a sort of debt crisis.

The analysis of these macroeconomic variables in isolation are not insightful. However, looking at the broader macroeconomic picture provides insight into the current conditions for a business with prospects to invest further into the future. Knowing more about the actual environment is a type of forecasting, as the analysis of fundamental variables is important to having a strong support for the current levels of business confidence.

Text Mining

Having a description of each observation will help to provide insight into the meaning of the dataset. We have explained the process of text mining before in the methods section, so here we will go straight to the table results. They show the top 25 words for each subset of data. This subset of data is for California in the year 2018. The first table shows the top 25 weighted words coming from the business categories variable and the second table shows the same calculation for the awarding agency names variable.

busi	small	categori	educ	govern	higher	institut
0.272	0.128	0.128	0.093	0.077	0.047	0.047
region	state	nonprofit	own	corpor	entiti	exempt
0.037	0.037	0.019	0.016	0.016	0.016	0.016
tax	design	special	american	indian	nativ	tribal
0.016	0.015	0.015	0.001	0.001	0.001	0.001
good	manufactur	local	partnership			
0.001	0.001	0.000	0.000			

Looking first at the stem words from the business categories, we can see that there seems to be a real demand coming from small businesses. This may indicate that the small business economy in California is receiving large funding from the federal government. Having clues such as this could lead to further research to find out more information. The table also shows that higher education seems to be getting some significant contributions from the federal government. This information would help colleges to better understand that there is a great deal of funding going on now on their behalf. An interesting note is that Native American groups seem to be getting some funding as well, something that may be interesting for those special groups who need government assistance.

depart	energi	transport	defens	agricultur	interior	administr
0.493	0.283	0.11	0.049	0.043	0.004	0.003
nation	aeronaut	space	servic	health	human	labor
0.002	0.002	0.002	0.002	0.001	0.001	0.001
general	secur	homeland	commerc	develop	justic	agenc
0.001	0.001	0.001	0	0	0	0
intern	educ	sta	affair			
0	0	0	0			

The second table shows the top 25 weighted stem words coming from the funding agency. It seems that the top departments are those of energy, transport, defense, and agriculture. This would be good information for any business who is related to these areas of work and are looking for government funding for contractual work. They could possibly meet these departments to see if they can strike any interesting business deals which can benefit both the company's revenue along with the local government or communities.

Machine Learning

State	Logistic Regression	Support Vector Machine	K-means Clustering
IL	0.22 ± 0.01	0.18 ± 0.06	0.063
NJ	0.23 ± 0.02	0.15 ± 0.13	0.101
FL	0.18 ± 0.02	0.14 ± 0.08	0.081
NC	0.17 ± 0.05	0.14 ± 0.05	0.130
MD	0.16 ± 0.01	0.12 ± 0.04	0.058
MS	0.19 ± 0.03	0.14 ± 0.09	0.096
SD	0.16 ± 0.02	0.14 ± 0.06	0.065
AZ	0.17 ± 0.03	0.13 ± 0.07	0.107

The table shows some of the state we selected based on the most frequent appeared states in each quantile interval of state frequencies and the state total_obligation. As the result shown, the performance of these three models were bad. The correlation between the 3 variables (agency, sector, business category) we used and the total obligation is very weak. The highest accuracy we got was 23%. If we consider that the accuracy of random guesses is 10%, we can see that our models were only slightly better. Not to mention the amount of data we have and the time we spent on training.

Conclusion / Discussion

Given that the variable that we are trying to estimate and predict is the federal government spending, we were initially looking for parameters to best maximize this response variable. The dataset is immense, with 61 different variables to choose from, however, like many other large datasets there is an issue of finding variables which can be considered

seriously for the purpose of prediction. When looking at certain groups of variables, it is apparent that many of them are highly similar and so potentially only one of these are usable in our model. Other times, we encounter groups of variables which are not interesting for the purpose of our research.

The result of this apparent paradox of large datasets, we had decided to choose variables that made sense intuitively, based on their descriptions and appearance within the dataset. Despite this worry during the modeling process, we were still able to gain substantial insight into the available data given by the website 'usaspending.' The initial exploratory stages of analyzing the variables allowed us to paint a picture of the potential business environment for a business owner who is interested in finding some contractual work with the federal government.

Looking at the time series analysis, we were able to determine the forecasted federal spending by state annually. Such information would make it possible for a business owner to either begin investing or act wisely by saving. In our current case, it is apparent that a business owner should be weary of being too confident in their customer base. Throughout all the states the trend right now is largely negative in terms of growth and so uncertainty lies around the corner. If there were more time, it would have been worthwhile to extend the analysis towards monthly or weekly time series. This would involve using a variable different from the yearly data so that we would be allowed a larger sample size. The sample size in our case had been limited to less than 10 observations, a serious issue in time series. Therefore, it would have possibly been worth having a more complicated dataset for the purpose of better prediction.

Using Geoplotting again proves to be a worthwhile technique when it comes to data which has distinct geographic regions. The variable relating to state recipient information makes it possible to visualize in a manner the results that can't be understood merely by analyzing trends or reading statistics about the data. Seeing the geographic visualization brings in a much higher dimension of awareness about the data and makes it possible for new conclusions to be drawn. For example, looking at the east coast we are able to see a large difference in how prevalent the services industry is.

Seeking out patterns such as this makes it possible for interested parties to gravitate towards where they are needed and to find out where it is that they wish to be going towards. If there was more time, we would have liked to include the interactive Plotly Geoplots, however this time they would have included a time slide bar. Having such a feature would allow business owners to see the time trend for their relevant sector. This

information makes it possible for them to decide whether or not it is time to jump onto a growing trend, or even to relocate elsewhere that the trend is still alive and going strong.

The macroeconomic variables came in handy when it came towards developing a more fundamental understanding of the various factors that are evident in the dataset. The macroeconomic view makes it possible to understand the cyclical nature of what is taking place. When analyzing different variables and trying to understand the underlying momentum of where things are headed, it could be easy to start seeing things going up and down in a seemingly overall random pattern. When thinking about the big picture through a macroeconomic lens, it is possible to see that there is reason behind all the different changes taking place.

As a business owner, it is vital to have an understanding of these economic variables as they are directly related to the business owner themself. Such variables can help a business owner decided whether or not it is time for them to being investing their hard earned money into a potentially larger return or whether it is good to play it safe and keep on getting by until the economy picks up again. The macroeconomic visualizations along with a theoretical background for analysis makes it possible to understand a highly intricate landscape. Given more time and space, it would have been interesting to dig further into the relationship of recent economic events and how they can be best interpreted for a business owner. For example, we would like to explain the current pros and cons of investing in more capital as it is possible that the economy may be headed for a serious downturn before everything picks up again.

Text mining makes it possible for businesses to understand where they can possibly fit in. The variables describe both the types of businesses along with the funding agencies that are responsible for paying for the contracts to do work for the government. The text mining process utilizes the same technique which was learned earlier and similarly provides insightful information as to what sort of funding or businesses are prevalent in each state. This information can help business owners to identify times and places where they can best see themselves. If they understand that their work is not as popular as it once was, then they may consider restructuring their business so that they can keep up with changing times. Another business owner may in fact find out that they are better going elsewhere that they can be more profitable. If there was more time to do research on Natural Language Processing (NLP), it would have been worthwhile to explore more methods for understanding NLP techniques. An example would be utilizing word clouds to somehow interpret the same idea through a different methodology.

However, based on the results of different machine learning models, it seems that those variables are not representative of the the federal spending. Initiliatially we treat this problem as a prediction problem, but the results are too disappointing to be included. Although we changed a prediction problem into a 10 class classification problem in the end, the highest accuracy is only 23% among 3 three models we have tried. The results may improve if we have time to explore on more models, but it is a somehow strong indicator that other processes or parameters could be significantly improved. There maybe a better way to reduce the levels of those categorical data, and a better way to encode those levels. Further exploration could focus on those 2 aspects and try to incorporate variable that will be more deterministic. Right now, the model would be disappointing for business owners.

An application which we would have liked to take advantage of is the SQLite database which had been provided for use by students. It is apparent that it would be preferable to directly retrieve the statistics that are of interest from the database rather than downloading the dataset itself and working through the data structures to obtain certain calculations. Several issues came up early on during this process and not being able to work efficiently with the R packages available to handle the SQLite database made it an undesirable platform to work with. For that reason the large database was not utilized, despite the potential benefits of doing so.

It became apparent during the exploratory process that R would have a difficult time with such a large dataset. To overcome this issue, it became necessary to be mindful of the size of the data that is being programmed. It was necessary during one time to chunk through the data and to retrieve the subset of data that was of interest. Rather than being able execute a simple command to modify the data, it was necessary to use an ad-hoc method of carefully working with the data structures so that the key data could be maintained while the unnecessary information could be discarded.

After having learned to code in R for some time, it had become apparent that it was necessary to write in a style of functional programming. This makes it possible to mold data into different shapes that make it possible to generate interesting statistics that are important for the purposes of this project. However, when dealing with such a massive dataset, it is apparent that many nuances can lie hidden and not be made apparent until a fairly extensive and somewhat intricate system had been created. Therefore, in order to discover these occasional bugs it was important to carefully maintain different parts of code and to recognize patterns that work along with being able to realize when a new instance of some issue arises which has not been encountered before.

Appendix

Reference

https://stackoverflow.com/questions/14848172/appending-a-list-to-a-list-of-lists-in-r

https://www.r-bloggers.com/r-tip-use-stringsasfactors-false/

https://stackoverflow.com/questions/29522841/the-curious-case-of-arima-modelling-using-r

https://stackoverflow.com/questions/45439317/how-to-save-plots-in-r-and-have-them-nice-looking

https://stackoverflow.com/questions/40397833/is-there-a-way-to-prevent-astsasarima-from-plotting

https://www.statmethods.net/advgraphs/axes.html

https://stackoverflow.com/questions/25707647/ merge-multiple-spaces-to-single-space-remove-trailing-leading-spaces

https://www.analyticsvidhya.com/blog/2017/03/beginners-guide-on-web-scraping-in-r-using-rvest-with-hands-on-knowledge/

https://www.missourieconomy.org/about_us/naics_sect.stm

https://www.r-graph-gallery.com/215-the-heatmap-function/

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://scikit-learn.org/stable/modules/cross_validation.html

https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

https://stackoverflow.com/questions/38079853/how-can-i-implement-incremental-training-for-xgboost

https://stackoverflow.com/questions/4357101/promise-already-under-evaluation-recursive-default-argument-reference-or-earlie

https://stackoverflow.com/questions/41399795/savewidget-from-htmlwidget-in-r-cannot-save-html-file-in-another-folder

https://github.com/tidyverse/ggplot2/issues/1048

https://stackoverflow.com/questions/4862178/remove-rows-with-all-or-some-nas-missing-values-in-data-frame

https://stackoverflow.com/questions/14164525/splitting-a-large-data-frame-into-smaller-segments

https://stackoverflow.com/questions/15227887/how-can-i-subset-rows-in-a-data-frame-in-r-based-on-a-vector-of-values

https://stackoverflow.com/questions/5237557/extract-every-nth-element-of-a-vector

http://www.sthda.com/english/wiki/ggplot2-title-main-axis-and-legend-titles

https://stackoverflow.com/questions/19622063/adding-vertical-line-in-plot-ggplot

https://stackoverflow.com/questions/5106782/use-of-ggplot-within-another-function-in-r
jeffgswanson.com/create-us-heatmap-r

https://plot.ly/r/choropleth-maps/#choropleth-maps-in-r

https://cran.r-project.org/web/packages/fredr/vignettes/fredr.html

https://fred.stlouisfed.org/release?rid=140

https://ggplot2.tidyverse.org/reference/geom_abline.html

https://www.r-bloggers.com/how-to-summarize-a-data-frame-by-groups-in-r/

https://stackoverflow.com/questions/50935857/ggplot2-keeps-adding-5-at-end-of-year-variable-on-x-axis

https://stackoverflow.com/questions/10438752/adding-x-and-y-axis-labels-in-ggplot2

https://stackoverflow.com/questions/6142944/how-can-i-plot-with-2-different-y-axes

https://usaspending-help.zendesk.com/hc/en-us/community/posts/115006369387-Why-are-some-total-obligations-negative-,

https://usaspending-help.zendesk.com/hc/en-us/community/posts/360021405654-Recipient-Business-Type-Correction

https://stackoverflow.com/questions/24361598/reduce-size-of-legend-area-in-barplot

Code

proj.R

```
library(dplyr); library(ggplot2); library(astsa); library(rvest) # Load
  libraries
library(htmlwidgets); library(RColorBrewer); library(fredr)
library(lubridate); library(tidyverse); library(plotly); library(scales)
library(beepr); library(tm); library(SnowballC); library(NLP)

# Load/Clean Data
transaction <- read.csv('transaction.csv', stringsAsFactors = FALSE)
data_sub <- subset(transaction, select =
  c('recipient_location_state_code',
  'total_obligation', 'award_id', 'fiscal_year'),
  transaction$recipient_location_state_code %in% state.abb &
  transaction$total_obligation > 0 &
  transaction$fiscal_year > 2010 &
  transaction$fiscal_year < 2019)

data_list <- split(data_sub, (seq(nrow(data_sub))-1) %/% 10000)

unique_id_func <- function(list_element) {
  # This function works to get the unique total_obligation per
  # award_id. It does so using the following methods:
  # Using the list of dataframes which have been split by a certain
  # length, split again each dataframe by award_id. It then coalesces
  # each of these award_id's such that all the repeated information
  # is removed and NA's are ignored if possible.
  id_split <- split(list_element, list_element$award_id)
  # head(id_split)
  # id_split <- id_split[1:2]
  df_list <- lapply(id_split, function(x) lapply(1:nrow(x), function(y)
  x[y,]))
  coalesce_list <- lapply(df_list, function(x) coalesce(!!!x))
  do.call('rbind', coalesce_list)
}

# Merge data together and repeat for final data frame
data_list_merge <- lapply(data_list, unique_id_func)
data_list_comb <- do.call('rbind', data_list_merge)
data_list_award_table <- table(data_list_comb$award_id)
```

```
data_list_award_table_sub <- data_list_award_table[data_list_award_table >
  1]
remain_dupl <- names(data_list_award_table_sub)
remain_dupl_df <- data_list_comb[data_list_comb$award_id %in% remain_dupl,
  ]
not_dupl_df <- data_list_comb[!data_list_comb$award_id %in% remain_dupl, ]
last_coalesce <- unique_id_func(remain_dupl_df)
data_comb <- rbind(last_coalesce, not_dupl_df)
data_comb$recipient_location_state_code <-
  as.character(data_comb$recipient_location_state_code)

# spending by state
ann_state_spending <- data_comb %>%
  select(recipient_location_state_code, total_obligation, fiscal_year) %>%
  group_by(fiscal_year, recipient_location_state_code) %>%
  summarise(state_spending = sum(total_obligation))

ann_state_spending <- as.data.frame(ann_state_spending)

### Plotting Spending
make_lineplot = function(folder_path, state_index, mydata) {
  # Create barplot of state spending
  png(paste(folder_path, state.abb[state_index], ".png", sep = '')) # Save
  to file
  myplot <- ggplot(data = subset(mydata, recipient_location_state_code ==
                                   state.abb[state_index]), aes(x =
  fiscal_year, y = state_spending)) +
    geom_line(aes(y = state_spending, colour = 'State Spending')) +
    geom_line(aes(y = ann_spending$ann_sum/50, colour = 'Country
  Spending')) +
    scale_x_continuous(breaks = 0:2100) +
    xlab(paste('Year')) + ylab('Annual Spending') +
    ggtitle(paste(state.name[state_index], ' Annual Spending')) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_y_continuous(sec.axis = sec_axis(~., name = 'Country'))
  print(myplot)
  dev.off()
}
sapply(1:length(state.abb), function(x) # Create barplot for all states
  make_lineplot(folder_path = "./state_spending/",
                state_index = x,
                mydata = ann_state_spending))

state_national_spending_func <- function(annual_state_spending =
  ann_state_spending,
                                         annual_national_spending =
  ann_spending,
                                         state_index = 1, folder_path) {
  # Plot the GDP for the whole country With local state GDP.
  png(paste(folder_path, state.abb[state_index], ".png", sep = ''),
```

```
      width = 1920, height = 1000) # Save to file
  time <- seq(2011, 2018, 1)
  left_y_axis <-
  annual_state_spending[annual_state_spending$recipient_location_state_code
  ==
                                          state.abb[state_index],]
  right_y_axis <- annual_national_spending$ann_sum
  par(mar=c(5, 4, 4, 6) + 0.1)

  plot(time, left_y_axis$state_spending, pch=16, axes=FALSE, xlab="",
  ylab="",
       type="b",col="black", main = paste0(state.name[state_index], '
  Annual Spending'))
  axis(2, ylim=c(range(left_y_axis$state_spending)
  [1],range(left_y_axis$state_spending)[2]),
       col="black",las=1)  ## las=1 makes horizontal labels
  mtext(paste0(state.abb[state_index], ' Spending'),side=2,line=2.5)
  box()
  par(new=TRUE)

  ## Plot the second plot and put axis scale on right
  plot(time, right_y_axis, pch=15,  xlab="", ylab="",
       axes=FALSE, type="b", col="red")
  ## a little farther out (line=4) to make room for labels
  mtext("National Spending",side=4,col="red",line=4)
  axis(4, col="red",col.axis="red",las=1)

  ## Draw the time axis
  axis(1,pretty(range(time)))
  mtext("Years",side=1,col="black",line=2.5)

  ## Add Legend
  legend("topright",legend=c(paste0(state.abb[state_index], ' Spending'),
                             "National Spending"),
  text.col=c("black","red"),pch=c(16,15),col=c("black","red"))
  dev.off()
}

sapply(1:length(state.abb), function(x)
  state_national_spending_func(folder_path = "./state_spending/",
                               state_index = x))

### Time Series Forecasting
norm_diff_plots <- function(state_spending = ann_state_spending,
  curr_state = state.abb,
                            state_index = 1, folder_path) {
  png(paste(folder_path, curr_state[state_index], ".png", sep = ''),
      width = 1920, height = 1000) # Save to file
  par(mfrow=c(2,2))
```

```
  curr_state_spending <-
  state_spending[state_spending$recipient_location_state_code ==
                                          curr_state[state_index], ]
  plot(curr_state_spending$fiscal_year,
  curr_state_spending$state_spending, type = 'l',
       main = paste(curr_state[state_index],' Total Obligation'),
       xlab = 'Year', ylab = 'Annual Total Obligation')
  points(curr_state_spending$fiscal_year,
  curr_state_spending$state_spending)

  # Differenced data
  plot(curr_state_spending$fiscal_year[-1],
  diff(curr_state_spending$state_spending),
       type = 'l', main = paste(curr_state[state_index],' Differenced
  Total Obligation'),
       xlab = 'Year', ylab = 'Differenced Annual Total Obligation')
  points(curr_state_spending$fiscal_year[-1],
  diff(curr_state_spending$state_spending))

  # ACF/PACF
  acf(diff(curr_state_spending$state_spending), main = 'ACF
  (differenced)')
  pacf(diff(curr_state_spending$state_spending), main = 'PACF
  (differenced)')
  dev.off()
}

sapply(1:length(state.abb), function(x)
  norm_diff_plots(state_spending = ann_state_spending, state_index = x,
                  folder_path = './forecast/'))

# Time Series modeling
ann_spending_scaled <- ann_state_spending
ann_spending_scaled$state_spending <-
  log(ann_spending_scaled$state_spending)

fitting_sarima = function(i, j, state_forecast = curr_forecast){
  sarima(state_forecast$state_spending,
         p = i, d = 0, q = j, Model = FALSE, details = FALSE)
}

ann_spending_forecast <- function(ann_spending = ann_spending_scaled,
  folder_path,
                                  curr_state = state.abb, state_index = 1)
  {
  # This function will plot the observed and fitted values for an ARIMA
  model according to AICc.
  # Additionally, it will create a forecast for the next 5 years in annual
  total obligation.

  png(paste(folder_path, state.abb[state_index], ".png", sep = ''))
```

```
  # Load current state data, and find the p, q for ARIMA(p,1,q)
  curr_forecast <- ann_spending[ann_spending$recipient_location_state_code
  ==
                                  curr_state[state_index],]
  pq_choices <- data.frame(p = rep(0:4, each = 5), q = rep(0:4, 5))
  all_sarima <- mapply(fitting_sarima, pq_choices$p, pq_choices$q)

  npair = dim(all_sarima)[2]
  AICC_result = sapply(1:npair, function(x) all_sarima[,x]$AICc)
  pq <- pq_choices[which.min(AICC_result),]

  arima_p1q <- sarima(curr_forecast$state_spending, # Fit ARIMA(p,1,q)
  model
                      p = pq[,1], d = 1, q = pq[,2],
                      details = FALSE, Model = FALSE)

  sarima.for(curr_forecast$state_spending, n.ahead = 5,
             p = pq[,1], d = 1, q = pq[,2])
  title(paste(state.name[state_index], ' Annual Spending Forecast'))
  dev.off()
}

sapply(1:length(state.abb), function(x)
  ann_spending_forecast(folder_path = './arima_forecast/', state_index =
  x))

### Sector Analysis
### Web scrape NAICS sector information
url <- 'https://www.missourieconomy.org/about_us/naics_sect.stm'
webpage <- read_html(url)
html_nodes(webpage,'table')
web_data <- webpage %>%
  html_nodes("table") %>%
  html_nodes("td") %>%
  html_text("td")

# Sort html table data
code_posi <- grep("[0-9]", web_data)
x <- code_posi[1]:length(web_data)
pos <- code_posi - 3
pat <- rep(seq_along(pos), times = diff(c(pos, length(x) + 1)))
row_num_posi_list <- split(x, pat)
naics_sec <- do.call("rbind", lapply(row_num_posi_list, function(x)
  web_data[x]))
wrong_sec_info <- grepl("[0-9]", naics_sec[,3])
naics_sec_corrected <- naics_sec[,3]
for (i in 1:length(naics_sec[,3])){ # Correct sector information
  if (wrong_sec_info[i]) {
```

```
    naics_sec_corrected[i] <- naics_sec_corrected[i-1]
  }
}
naics_sec[,3] <- naics_sec_corrected
naics_sec <- data.frame(naics_sec)

# Clean NAICS data frame
names(naics_sec) <- c("naics_code", "description", "sector")
naics_sec$description <- gsub("[\r\n\t]", "", naics_sec$description)
naics_sec$description <- gsub("\\s+", " ", naics_sec$description)
naics_sec$naics_code <- as.character(naics_sec$naics_code)
f1 <- rbind(naics_sec[4,], naics_sec[4,])
f1$naics_code <- c(31,32)
f2 <- rbind(naics_sec[6,], naics_sec[6,])
f2$naics_code <- c(48,49)
f3 <- rbind(naics_sec[8,], naics_sec[8,])
f3$naics_code <- c(44,45)
naics_sec <- rbind(naics_sec, f1, f2, f3)
naics_sec <- naics_sec[c(-4,-6,-8), ]
naics_sec_sub <- naics_sec[,-2]
rownames(naics_sec_sub) <- NULL

# Subset data to important sections
eda_states = subset(transaction, select =
  c('recipient_location_state_code',
  'fiscal_year', 'naics_code'),
  !is.na(transaction$naics_code) &
  transaction$recipient_location_state_code %in% state.abb &
  transaction$fiscal_year != "")

eda_states$naics_code_sub <- substr(eda_states$naics_code, start = 1, stop
  = 2)
eda_states$sectors <-
  naics_sec_sub$sector[match(eda_states$naics_code_sub,
  naics_sec_sub$naics_code)] # Add sector information

state_sec_func <- function(state_data = eda_states, state_index = 1) {
  # Find proportion of transactions by sector, returns percentage
  # of each sector per state.
  curr_state <- state_data[state_data$recipient_location_state_code ==
                             state.abb[state_index], ]
  table(curr_state$sectors) / sum(table(curr_state$sectors))
}

# Plotly
geo_usa <- list( # Plotly formatting for geoplot
  scope = 'usa',
  showland = TRUE,
  landcolor = toRGB("gray95"),
```

```
  countrycolor = toRGB("gray80")
)

sector_geo <- function(geo_plot = geo_usa, sector_df = services_df,
  color_scheme = 'Reds',
                       legend_title = "Percentage (%)", sec_name) {
  # Saves a plotly geo heatmap for a specific sector. Outlines the
  different
  # percentages that each state has for a given sector. Saves to
  interactive html.
  file_path <- file.path(getwd(), "sector_geo", paste0(sec_name, '.html'))
  sector_plot <-
    plot_geo(sector_df, locationmode = 'USA-states') %>%
    add_trace(
      z = sector_df$percentage, locations = sector_df$states,
      colors = color_scheme
    ) %>%
    colorbar(title = legend_title) %>%
    layout(
      title = paste('Prevalence of ', sec_name, ' Sector by State (by
  %)'),
      geo = geo_plot
    )
  saveWidget(as_widget(sector_plot), file = file_path)
  sector_plot
}
sector_geo(sector_df = services_df, sec_name = 'Services')

### Economic Data
api.key <- '65e973606fb36ebc54acd85cfe427fa9' # Load API
fredr_set_key(api.key)

# Load data from Economic Data from Fred St. Louis
fred_api <- function(series_id_start, series_id_end) {
  # The function uses the API to gather all the data by state from the
  # economic website data Fred St. Louis. The function will first loop
  # through the states to apply the correct series_id to each query.
  # Then, the function will turn clean the data structure, as well as
  # doing some data cleaning to make the data more simple.
  # input: series_id_start: the begining of the series_id, series_id_end:
  # the end of the series_id
  # output: dataframe of state data for each API query
  fred_list <- lapply(state.abb, function(state) fredr(series_id =
  paste(series_id_start,

  state, series_id_end, sep = '')))
  fred_df <- as.data.frame(do.call("rbind", fred_list))
  fred_df$series_id <- substr(fred_df$series_id, start =
  nchar(series_id_start) + 1,
```

```
                              stop = nchar(fred_df$series_id) -
  nchar(series_id_end))
  fred_df$date <- as.numeric(substr(fred_df$date, start = 1, stop = 4))
  return(fred_df)
}

NGSP <- fred_api(series_id_start = '', series_id_end = 'NGSP')

gdp_barplot <- function(folder_dir, state_index, gdp_data) {
  # Plot the GDP data
  png(paste(folder_dir, state.abb[state_index], ".png", sep = ''))
  myplot <- ggplot(data = subset(gdp_data, series_id ==
  state.abb[state_index]),
                   aes(x = date, y = value)) + geom_bar(stat = 'identity')
  +
    scale_x_continuous(breaks = 0:2100) +
    xlab(paste('Year')) + ylab('Annual GDP') +
    ggtitle(paste(state.name[state_index], ' Annual GDP')) +
    geom_vline(xintercept = 2007, color = 'red') +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  print(myplot)
  dev.off()
}

sapply(1:length(state.abb), function(x)
  gdp_barplot(folder_dir = "./state_gdp/", state_index = x, gdp_data =
  NGSP))

year_breaks_func <- function(year_vec, num_breaks) {
  # This function is for efficiently finding the number
  # of breaks to place inbetween the years on the x-axis.
  unique_years <- sort(unique(year_vec))
  unique_years[seq(1, length(unique_years), num_breaks)]
}

econ_ggplot <- function(econ_data, x_label = 'Years', y_label,
                        main_title, breaks_vec) {
  # Plot the economic data
  ggplot(econ_data) + geom_line(mapping = aes(x = date, y = value)) +
    facet_wrap(~ series_id, scales = 'free') + ggtitle(main_title) +
    xlab(x_label) + ylab(y_label) + geom_vline(xintercept = 2007) +
    theme(axis.text.x = element_text(angle = 30, hjust = 1)) +
    scale_x_continuous(breaks = breaks_vec)
}

econ_ggplot2 <- function(econ_data, x_label = 'Years', y_label,
                         main_title) {
  # Plot the economic data
  ggplot(econ_data) + geom_line(mapping = aes(x = date, y = value)) +
```

```
    ggtitle(main_title) + xlab(x_label) + ylab(y_label) +
    geom_vline(aes(xintercept = as.numeric(ymd('2007-02-20')))) +
    theme(axis.text.x = element_text(angle = 30, hjust = 1)) #+
}

econ_filter1 <- function(series_id_start = '', series_id_end = '',
  start_date = 0,
                         num_breaks, y_label, main_title) {
  # Filter the economic data so it can be plotted.
  econ_data <- fred_api(series_id_start = series_id_start,
                        series_id_end = series_id_end)
  econ_data <- econ_data[econ_data$date > start_date, ]
  econ_years <- year_breaks_func(year_vec = econ_data$date, num_breaks =
  num_breaks)

  econ_ggplot(econ_data = econ_data, y_label = y_label,
              main_title = main_title,
              breaks_vec = econ_years)
}

econ_filter2 <- function(series_id, start_date = 0,
                         y_label, main_title) {
  # Filter the economic data so it can be plotted.
  api_query <- fredr(series_id = series_id)
  econ_data <- api_query[, c('date', 'value')]
  econ_data <- econ_data[year(econ_data$date) > start_date, ]
  econ_ggplot2(econ_data = econ_data, y_label = y_label,
               main_title = main_title)
}

# pop size
state_pop <- econ_filter1(series_id_end = 'POP', start_date = 2000,
                          num_breaks = 3, y_label = 'Population',
                          main_title = 'State Population 2000-2018')

# med income
state_med_inc <- econ_filter1(series_id_start = 'MEHOINUS',
  series_id_end = 'A672N', start_date = 1995, num_breaks = 4,
  y_label = 'Median Income', main_title = 'State Median Income')

# education level
state_edu <- econ_filter1(series_id_start = 'GCT1502', num_breaks = 2,
  y_label = 'Percentage with Bachelor\'s Degree or Higher',
  main_title = 'State Education Level')

# unemployment
unemp_plot <- econ_filter2(series_id = 'UNRATE', start_date = 2000,
  y_label = 'Unemployment (%)', main_title = 'Unemployment Rate')
```

```r
# interest rates
ir_plot <- econ_filter2(series_id = 'FEDFUNDS', start_date = 1980,
  y_label = 'Interest Rate (%)', main_title = 'Interest Rates')

# debt
debt_plot <- econ_filter2(series_id = 'GFDEBTN', start_date = 1970,
  y_label = 'Millions ($)', main_title = 'Total Public Debt (1970-2018)')

### Text Mining
bus_cat <- colnames(transaction)[c(5, 6, 8, 36, 61)]
award_agency <- colnames(transaction)[c(5, 6, 8, 36, 53)]

# Initial subset
tran_sub <- subset(transaction, select = bus_cat,
  transaction$fiscal_year > 2010 & transaction$fiscal_year < 2019 &
  transaction$total_obligation > 0 & transaction$business_categories != ''
 &
  transaction$recipient_location_state_code %in% state.abb &
  transaction$award_id != '')

tran_sub2 <- subset(transaction, select = award_agency,
  transaction$fiscal_year > 2010 & transaction$fiscal_year < 2019 &
  transaction$total_obligation > 0 &
  transaction$awarding_toptier_agency_name != '' &
  transaction$recipient_location_state_code %in% state.abb &
  transaction$award_id != '')

### text_mine() functions
tran_dupl_removal <- function(tran_data) {
  # Combine duplicate award_id's
  id_table <- table(tran_sub$award_id)
  id_table_sub <- id_table[id_table > 1]
  dupl_id <- names(id_table_sub)
  tran_dupl <- tran_sub[tran_sub$award_id %in% dupl_id, ]
  tran_not_dupl <- tran_sub[!tran_sub$award_id %in% dupl_id, ]
  tran_dupl_unique <- unique_id_func(tran_dupl)
  tran_comb <- rbind(tran_not_dupl, tran_dupl_unique)
  return(tran_comb)
}

year_to_state <- function(yearly_df = tran_data_year[[1]], desc =
  'business_categories') {
  # Returns two lists, where the description and spending for a specific
  # year for all states is returned.
  yearly_df <- data.frame(yearly_df, stringsAsFactors = FALSE)
  state_split <- split(yearly_df, yearly_df$recipient_location_state_code)
  state_desc <- lapply(state_split, function(x) x[,desc])
  state_spend <- lapply(state_split, function(x) x$total_obligation)
  return(list(state_desc, state_spend))
```

```
}

annual_desc_spend <- function(year_list_element =
  year_to_state_desc_spend[[1]]) {
  # Takes the yearly list element and breaks down into list of 50
  descriptions
  # and 50 spending amounts. Then filters the descriptions into a weighted
  # character vector. Returns top 25.
  year_desc <- year_list_element[[1]]
  year_spend <- year_list_element[[2]]

  # Go by state to find the weighted character vectors.
  # Creates a list of 50 states each with a list of an index and string.
  state_desc_word_filter <- lapply(year_desc,
                                   function(x)
  word_filter(year_desc_element = x))
  # Add the spending to the list of 50.
  for(i in 1:length(state_desc_word_filter)) {
    state_desc_word_filter[[i]][length(state_desc_word_filter[[i]]) + 1]
  <-
      year_spend[i]
  }

  # Sort through each 50 states in list and use the word filter.
  state_weighted_top25 <- lapply(state_desc_word_filter, function(x)
    state_list_to_char_vec(state_string_ind_spend = x))
  return(state_weighted_top25)
}

word_filter <- function(year_desc_element = year_desc[[1]]) {
  # Filters words to stem words.
  # Process of doing initial text mining of description.
  removed_punc = gsub('[[:punct:]]', ' ', year_desc_element)
  lower_case = tolower(removed_punc)

  # Remove stopwords, create stem words
  stop_words1 = removeWords(lower_case, stopwords('en'))
  stop_words2 = removeWords(stop_words1, stopwords('SMART'))
  stem_words = stemDocument(stop_words2)

  # Deal with any white space and create removal index
  clean_whitespace = stripWhitespace(stem_words)
  remove_na = clean_whitespace[clean_whitespace != 'NA']
  rm_ind = which(remove_na != '')
  lead_trail = gsub("^\\s+|\\s+$", "", clean_whitespace[rm_ind])
  rm_ind = which(lead_trail != '')

  return(list(lead_trail, rm_ind))
}
```

```r
state_list_to_char_vec <- function(state_string_ind_spend =
                                     state_desc_word_filter[[1]]) {
  # Creates a data frame from filtered descriptions and then returns
  # the final top 25 weighted words.
  lead_trail <- state_string_ind_spend[[1]]
  rm_ind <- state_string_ind_spend[[2]]
  price <- state_string_ind_spend[[3]]

  character_vector <- data.frame(keywords = lead_trail[rm_ind], # df of
  keywords and prices
                                 weighted_price = price[rm_ind] /
                                   sapply(strsplit(lead_trail[rm_ind], "
  "), length),
                                 stringsAsFactors = FALSE)
  weighted_char_vectors(price = price[rm_ind], character_vector =
  character_vector)
}

weighted_char_vectors <- function(price, character_vector) {
  # The weighted_char_vectors function will take in the price and
  character
  # vector which have previously been created. It will then split the
  # character vector into the individual keywords of each expense, and
  then
  # assign the average price to each keyword. These keywords are then
  combined
  # so that keywords which are matching will also have their weighted
  prices
  # from across different expenses summed together.

  # Split the keywords from each exepense, and assign them to their
  average
  # prices from each character vector.
  keywords_list = strsplit(character_vector$keywords, ' ', fixed = TRUE)
  keywords_list_len = sapply(keywords_list, length)
  avg_price = character_vector$weighted_price / keywords_list_len
  weighted_words = Map(cbind, keywords_list, avg_price)
  weighted_words = do.call("rbind", weighted_words)

  # Turn the keywords into a dataframe, and then group the dataframe
  # by keyword to sum the total weighted average.
  weighted_df = as.data.frame(weighted_words, stringsAsFactors = FALSE)
  names(weighted_df) = c("keywords", "char_weights")
  weighted_df$char_weights = as.numeric(weighted_df$char_weights)
  sum_df = tapply(weighted_df$char_weights, weighted_df$keywords,
                  function(x) sum(x, na.rm = TRUE))

  # Return the top 25 keywords, along with their weighted prices in terms
  # of their proportion.
```

```
  top_25 = head(sort(sum_df, decreasing = TRUE), n = 25)
  round(top_25 / sum(top_25), 3)
}

text_mine <- function(data_sub = tran_sub, curr_desc =
  'business_categories') {
  # Collects the top 25 stem words from each state by year/
  data_sub$recipient_location_state_code <- # Fix datatype
    as.character(data_sub$recipient_location_state_code)
  data_sub <- data.frame(data_sub, stringsAsFactors = FALSE)

  tran_comb <- tran_dupl_removal(tran_data = data_sub) # Remove duplicates

  tran_data_year <- split(tran_comb, tran_comb$fiscal_year) # Split by
  year

  year_to_state_desc_spend <- lapply(tran_data_year,
                                     function(x) year_to_state(yearly_df =
  x, desc = curr_desc))

  ann_state_desc <- lapply(year_to_state_desc_spend, function(x)
    annual_desc_spend(year_list_element = x))
  ann_state_desc
}

# Create weighted character vectors
bus_text_mine <- text_mine(data_sub = tran_sub,
                           curr_desc = business_categories)
agency_name_text_mine <- text_mine(data_sub = tran_sub2,
                                   curr_desc =
  awarding_toptier_agency_name)
```

xgb.py

```
import pandas as pd
import numpy as np
import xgboost as xgb

# read in the data
train_file = './data/training.csv'
test_file = './data/testing.csv'
chunksize = 100000
train_set = pd.read_csv(train_file, chunksize=chunksize)
test_set = pd.read_csv(test_file)

# create the evaluation data
dtest = xgb.DMatrix(data=test_set.drop(['money'], axis=1),
  label=test_set['money'])
```

```
# set up the parameters for the model
params = {'max_depth': 6, 'learning_rate': 0.1, 'min_child_weight': 3}
model = None

# adding new training data by chunk
for i, batch in enumerate(train_set):
    print('{} iterations'.format(i))

    dtrain = xgb.DMatrix(data=batch.drop(['money'], axis=1),
  label=batch['money'])

    watchlist = [(dtest, 'eval'), (dtrain, 'train')]

    model = xgb.train(params, dtrain, 10, evals=watchlist,
  verbose_eval=True, xgb_model=model)

model.save_model('xgb_model')
```

clf_models.py

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.cluster import KMeans
from sklearn import metrics

selected_state = [1,6,9,11,20,30,32,37]
result = {}

# read and convert the columns to categorical
print('reading')
data = pd.read_csv('./data/shuffled_labeled.csv')
data['state'] = data['state'].astype('category')
data['agency'] = data['agency'].astype('category')
data['naics'] = data['naics'].astype('category')
data['business'] = data['business'].astype('category')
data['label'] = data['label'].astype('category')

# train the model for each state
for state in selected_state:

    subset = data[data['state'] == state]

    x = subset.drop(['money', 'label', 'state', 'award_id'], axis=1)
    y = subset['label']
```

```
    print(state)
    print('fitting with logistic...')
    log = SGDClassifier(loss='log', max_iter=5)
    scores = cross_val_score(log, x, y, cv=5)
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
2))



    print('fitting with SVM GD...')
    svm = SGDClassifier(loss='hinge', max_iter=10, n_jobs=-1)
    scores = cross_val_score(svm, x, y, cv=5)
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
2))



    kmeans = KMeans(n_clusters=10, n_jobs=-1).fit(x)
    score = metrics.normalized_mutual_info_score(y, kmeans.labels_)
    print("Kmeans - {}".format(score))
```

dummies_submit.sh

```
#!/bin/bash -l

# Use the staclass partition. Only applies if you are in STA141C
#SBATCH --partition=staclass

# Use two cores to get some pipeline parallelism
#SBATCH --cpus-per-task=2

# Give the job a name
#SBATCH --job-name=dum

#SBATCH --mail-type=ALL
#SBATCH --mail-user=shsiao@ucdavis.edu

srun python get_dummies.py
```

get_dummies.py

```
import pandas as pd
import numpy as np

filename = './data/1_shuffle.csv'
train = pd.read_csv(filename)
```

```
def process_batch(batch):
    # transform to categorical variable
    state_dummies = pd.get_dummies(batch['state'], prefix='state')
    print('state finished')
    agency_dummies = pd.get_dummies(batch['agency'], prefix='agency')
    print('agency finished')
    naics_dummies = pd.get_dummies(batch['naics'], prefix='naics')
    print('naics finished')
    business_dummies = pd.get_dummies(batch['business'],
  prefix='business')
    print('business finished')


    # put everything back together
    x = pd.concat(
        [state_dummies, agency_dummies, naics_dummies, business_dummies,
  batch['money']],
        axis=1)
    print('concat completed')


    x.to_csv('./data/with_dummies_shuffle.csv', index=False)

process_batch(train)
```

8_organized.R

```
library(data.table)
library(tidyr)
library(dplyr)
library(plyr)

#Read files
path = "~/Desktop/sta141c/final project/"
setwd(path)
names = c("fiscal_year","award_id","money")
file_money = fread("8.csv",col.names = names, na.strings = c("","  ", "NA",
  "NAN"))
colnames(file_money) = names
temp = data.frame(file_money$award_id, file_money$money)
file_money$money = NULL
colnames(temp) = c("award_id", "money")

#process the file
(sum(is.na(temp$money)) + sum(temp$money < 0, na.rm = TRUE)) /
  length(temp$money)
temp = temp[order(temp$award_id, temp$money),]
temp = temp[!duplicated(temp$award_id),]
```

```
file_money = file_money[order(file_money$award_id,
  file_money$fiscal_year),]
file_money = file_money[!duplicated(file_money$award_id),]
file_money$money = temp$money
file_money = na.omit(file_money)
file_money = file_money[file_money$money >= 0,]

#adjust inflation for total obligation
#2011,2012,2013,2014,2015,2016,2017,2018
inflation = c(1.03,1.017,1.015, 1.008, 1.007, 1.021, 1.021, 1.019)
file_money$money = file_money$money / inflation[file_money$fiscal_year -
  2010]
file_money$fiscal_year = NULL
write.csv(file_money, "col8.csv",  row.names = FALSE)
```

36_organized.R

```
library(data.table)
library(tidyr)
library(dplyr)
library(plyr)

# Reads file
path = "~/Desktop/sta141c/final project/36"
setwd(path)
names = c("fiscal_year","award_id","state")
file_state = fread("36.csv",col.names = names, na.strings = c("","  ",
  "NA", "NAN", "00"))
colnames(file_state) = names
#Order states, and get frequency table
file_state = file_state[order(file_state$award_id, file_state$state),]
state_result = file_state[!duplicated(file_state$award_id),]
state_result$state[is.na(state_result$state)] = "NA"
state_result$state = mapvalues(state_result$state, c("GEORGIA",
  "VIRGINIA"), c("GA", "VA"))
prop_table = prop.table(table(state_result$state))
prop_table = prop_table[order(prop_table, decreasing = TRUE)]
head(prop_table)
#Encode the state mapping
mapping_key = rownames(prop_table)
mapping_value = c(1:40, rep(41, length(mapping_key) - 40))
state_result$state = mapvalues(state_result$state, mapping_key,
  mapping_value)
state_encoding = data.frame(mapping_key, mapping_value)
write.csv(state_encoding, "state_encoding.csv")
write.csv(state_result, "col36.csv", row.names = FALSE)
```

41_organized.R

```r
library(data.table)
library(tidyr)
library(dplyr)
library(plyr)
#Read file
path = "~/Desktop/sta141c/final project/"
setwd(path)
names = c("fiscal_year","award_id","naics")
file_naics = fread("41.csv",col.names = names, na.strings = c("","  ",
  "NA", "NAN"))
colnames(file_naics) = names
#process file and get frequency table
file_naics$naics = as.character(file_naics$naics)
file_naics$naics = substr(file_naics$naics,1,2)
file_naics = file_naics[order(file_naics$award_id, file_naics$naics),]
naics_result = file_naics[!duplicated(file_naics$award_id),]
naics_result$fiscal_year = NULL
naics_result$naics[is.na(naics_result$naics)] = "NA"
naics_result$naics = mapvalues(naics_result$naics, c("48", "44", "32",
  "31"), c("49", "45", "33", "33"))
prop_table = prop.table(table(naics_result$naics))
prop_table = prop_table[order(prop_table, decreasing = TRUE)]
head(prop_table)
sum(prop_table[1:7])
#encode the mapping
mapping_key = rownames(prop_table)
mapping_value = c(1:7, rep(8, length(mapping_key) - 7))
naics_result$naics = mapvalues(naics_result$naics, mapping_key,
  mapping_value)
naics_encoding = data.frame(mapping_key, mapping_value)
write.csv(naics_encoding, "naics_encoding.csv")
write.csv(naics_result, "col41.csv", row.names = FALSE)
```

53_organized.R

```r
library(data.table)
library(tidyr)
library(dplyr)
library(plyr)
library(parallel)

path = "~/Desktop/sta141c/final project/"
setwd(path)
#read files and process file
names = c("fiscal_year","award_id","agency")
file_agency = fread("53.csv",col.names = names, na.strings = c("","  ",
  "NA", "NAN"))
```

```
colnames(file_agency) = names
file_agency = file_agency[order(file_agency$award_id,
  file_agency$agency),]
file_agency$fiscal_year = NULL
agency_result = file_agency[!duplicated(file_agency$award_id),]
agency_result$agency[is.na(agency_result$agency)] = "NA"
prop_table = prop.table(table(agency_result$agency))
prop_table = prop_table[order(prop_table, decreasing = TRUE)]
#encode the agency
mapping_key = rownames(prop_table)
mapping_value = c(1:12, rep(13, length(mapping_key) - 12))
agency_result$agency = mapvalues(agency_result$agency, mapping_key,
  mapping_value)
agency_encoding = data.frame(mapping_key, mapping_value)
write.csv(agency_encoding, "awarding_agency_toptier_encoding.csv")
write.csv(agency_result, "col53.csv", row.names = FALSE)
```

61_organized.R

```
library(data.table)
library(tidyr)
library(dplyr)
library(plyr)
library(parallel)

#Read file
path = "~/Desktop/sta141c/final project/"
setwd(path)
names = c("fiscal_year","award_id","business")
file_business = fread("61.csv",col.names = names, na.strings = c("","  ",
  "NA", "NAN"))
colnames(file_business) = names
#Order file and get frequency table
file_business = file_business[order(file_business$award_id,
  file_business$business),]
file_business$fiscal_year = NULL
result_business = file_business[!duplicated(file_business$award_id),]
result_business$business[is.na(result_business$business)] = "NA"
result_business$business[result_business$business == ""] = "NA"
temp = mclapply(strsplit(result_business$business, " "), '[[', 1, mc.cores
  = detectCores())
temp = unlist(temp)
result_business$business = temp
unique(temp)
prop_table = prop.table(table(temp))
prop_table = prop_table[order(prop_table, decreasing = TRUE)]
head(prop_table)
#Encode the mapping
mapping_key = rownames(prop_table)
```

```
mapping_value = c(1:6, rep(7, length(mapping_key) - 6))
result_business$business = mapvalues(result_business$business,
  mapping_key, mapping_value)
business_encoding = data.frame(mapping_key, mapping_value)
write.csv(business_encoding, "business_encoding.csv")
write.csv(result_business, "col61.csv", row.names = FALSE)
```

label_money.R

```
library(data.table)

path = "~/Desktop/sta141c/final project/merge"
setwd(path)
df = fread("1.csv")
#encoding money by intervals
money_label = cut(df$money, breaks = quantile(f8$money, probs = seq(0, 1,
  0.1)),
                  include.lowest = TRUE, labels = 1:10)
df$label = money_label
write.csv(df, "money_labeled.csv", row.names = FALSE)
```

merge.R

```
library(data.table)
library(plyr)
library(tidyr)

file_path = "~/Desktop/sta141c/final project/merge/tmp"
setwd(file_path)
#Merge all file together
total = fread("col8.csv")
total$money = NULL
id_list = total$award_id
for( filename in list.files(getwd())) {
  file = fread(filename)
  file = file[file$award_id %in% id_list,]
  file$award_id = NULL
  total = cbind(total, file)
}

write.csv(total, "1.csv", row.names = FALSE)
```

shuffle.R

```r
library(data.table)

file_path = "~/Desktop"
setwd(file_path)
f1 = fread("1.csv")
f1_shuffle = f1[sample(nrow(f1)),]
write.csv(f1_shuffle,"1_shuffle.csv", row.names = FALSE)
```

state_quantile.R

```r
library(data.table)

path = "~/Desktop/sta141c/final project/"
setwd(path)
#Used to examine which state to use in ml
total = fread("1.csv")
colnames(total)
total$naics = NULL
total$agency = NULL
total$business = NULL
head(total)
money_dist_by_state = aggregate(total$money, by = list(unique.values =
  total$state), FUN = "sum")
money_dist_by_state = money_dist_by_state[order(money_dist_by_state$x),]
quantile(money_dist_by_state$x)

freq_dist_by_state = table(total$state)
freq_dist_by_state = prop.table(freq_dist_by_state)
freq_dist_by_state = freq_dist_by_state[order(freq_dist_by_state)]
quantile(prop.table())
```

cut_file.sh

```sh
DATAFILE="/scratch/transaction.zip"

unzip -p ${DATAFILE} |
    cut --delimiter=, --fields=5,6,36 |
    awk '$1 > 2010 && $1 < 2019' > 36.csv
unzip -p ${DATAFILE} |
    cut --delimiter=, --fields=5,6,61 |
    awk '$1 > 2010 && $1 < 2019' > 61.csv
unzip -p ${DATAFILE} |
    cut --delimiter=, --fields=5,6,53 |
    awk '$1 > 2010 && $1 < 2019' > 53.csv
unzip -p ${DATAFILE} |
```

```
    cut --delimiter=, --fields=5,6,41 |
    awk '$1 > 2010 && $1 < 2019' > 41.csv
unzip -p ${DATAFILE} |
    cut --delimiter=, --fields=5,6,8 |
    awk '$1 > 2010 && $1 < 2019' > 8.csv
```