Analysis of Health Factors' Effects on Health Outcomes in the United States

Abhijeet Royala, Ashritha Dandam, John Burns, Sonali Abhijit Patil, Whitney Walker

Indiana University-Purdue University, Indianapolis, IN 46202
aroyala@iu.edu, adandam@iu.edu, jolburns@iupui.edu, sopatil@iu.edu, wmwalker@iu.edu

Abstract

This project aims to discover relationships between health factors and health outcomes in the United States. Using models that account for social, physical environment and health services factors, we explore differences in these factors and two negative health outcomes across the United States. Additionally, we explore the possible relationships between these factors and outcomes, to help community members and policy makers understand what interventions in their area can have the most impact on their community's health.

Keywords

Public health, health outcomes, health factors, income, air quality, hospital quality, United States, low birthweight, premature death.

1. Project Scope

1.1 Introduction

Overview

Several factors contribute to the health of individuals and communities, including social factors, the physical environment, and health services (World Health Organisation, n.d). In order "to make an impact on improving health equity and providing more patient-centered care, it is necessary to better understand and address the underlying causes of poor health" (Anderman, 2016). There is evidence that these underlying causes may vary drastically in different regions (Swift, 2002). Therefore, it is imperative for policy makers and community members also understand how these causes correlate to health outcomes in their specific region. This project will analyze pollution (a physical environment health factor), income (a social health factor), and quality of hospital care (a health services factor) and how these three variables impact the health outcomes of regions in the United States.

Background on the Chosen Factors

- Income: The evidence of a positive correlation between individuals' socioeconomic status and their health outcome is extensive, with high-income individuals tending to be in better health than low-income persons (Larimore 2011). This data used for this project displays average income and is categorized by zip code.
- Quality of hospital care: By definition, the Medicare hospital quality ratings are based on myriad health outcomes, including mortality, safety of care, readmission, patient experience and effectiveness of care (Hospital Compare overall hospital rating, n.d.). The data used for this project gives quality ratings on a scale of 1-5 (5 being the best rating) for many hospitals in the United states.
- Pollution: "Air pollution is now fully acknowledged to be a significant public health problem, responsible for a growing range of health effects that are well documented from the results of an extensive research effort conducted in many regions of the world," (Kelly & Fussell, 2015).

Because of the acute effects air pollution can have on health outcomes, the data used in this project uses the standard national metrics for emissions: the Environmental Protection Agency's Toxic Release Inventory and Greenhouse Gas Reporting Inventory.

Background on Health Outcomes

Along with location, health outcomes are the central metric that connect all of our factors. The Center for Disease Control (CDC) defines positive health outcomes as "being alive; functioning well mentally, physically, and socially; and having a sense of well-being. Negative outcomes include death, loss of function, and lack of well-being. In contrast to these health outcomes, diseases and injuries are intermediate factors that influence the likelihood of achieving a state of health" (Parrish, 2010). This project's health outcomes data sources uses two health outcome metrics to determine the health outcomes of various regions in the United States: 1. premature death and 2. low birthweight. These two metrics align with the CDC recommendations: death and lack of well-being, respectively.

1.2 Aim

The aim of this research project is to map the relationship and correlation between various socioeconomic and environmental factors, including pollution, income, location, and hospital quality rating with health outcomes (premature death and low birthweight) by region in the United States (fip code and state).

Hypothesis

- Datasets will confirm that health outcomes and the effects of health factors differ by region.
- Datasets will show that by region: high income negatively correlates with negative health outcomes; high hospital quality ratings negatively correlate with negative health outcomes; and high pollution rates positively correlate with negative health outcomes.

Null Hypothesis

- Datasets will show that health outcomes and the effects of health factors do not differ in by region.
- Datasets will show that there are no significant relationships between health outcomes; or high income will not negatively correlate with negative health outcomes; high hospital quality ratings will not negatively correlate with negative health outcomes; and high pollution rates will not positively correlate with negative health outcomes.

1.3 Purpose

This project's purpose is to map individual determinants of health (average income, pollution levels and hospital quality ratings) with region (FIP code and state) and health outcomes (premature death and low birthweight). The project would like to confirm projected trends that high incomes negatively correlate with negative health outcomes (Larimore 2011); high pollution levels positively correlate with negative health outcomes (Kelly & Fussell, 2015); high hospital quality ratings negatively correlate with negative health outcomes (Hospital Compare overall hospital rating, n.d.). Additionally, the project would like to confirm that these factors and outcomes differ in a significant way based upon geographic location (Swift, 2002). If significant trends can be be identified, they can then be used to create more targeted and effective health outcome interventions.

2. Methodology

2.1 Steps of the Project

In this project, we focused on exploring how various health factors impact health outcomes. To do this, we utilized SQL, Python, Plotly, Jupyter, and phpMyAdmin to work with our data throughout four stages: data collection, data extraction and storage, data analysis, and data visualization. We followed the three-tier data architecture, with our SQL database being our data tier, python analysis being our logic tier, and visualizations being our presentation tier.

2.2 Original Team Members and Responsibilities

Name	Background	Skills/Experience	Primary role
Abhijeet Royala	Data science	No SQL or Python experience.	Research, support in technology
Annabel Butler	Data science master's student; worked as a processor with SQL	No Python experience, intermediate SQL experience	Research, support in technology
Ashritha Dandam	Pharmacy, health informatics.	No SQL or Python experience	Research, support in technology
John Burns	Radiology, health informatics. IT Project Manager.	SQL/Python/Project Management	Technology expert, project management support
Sonali Abhijit Patil	Doctor	Python,R programming and SQL experience.	Technology expert, support in research, linear regression expert
Whitney Walker	Health communications, public health, design, Science, Technology studies, life sciences.	No SQL or Python experience. Some project management experience.	Project management: proofing and editing; support in technology and research

2.3 Actual Team Members and Responsibilities

Our initial assessment of general responsibilities was fairly accurate. However, as the project progressed, the responsibilities became more specific to meet the group needs. The biggest changes to responsibilities occured because Annabel dropped the class. Her responsibilities had to be redistributed to other team members.

Name	Background	Skills/Experience	Primary role
Abhijeet Royala	Data science	No SQL or Python experience.	Research, support in analysis (normal distribution etc.)
Annabel Butler	*Data science master's student; worked as a processor with SQL*	*No Python experience, intermediate SQL*	*Dropped course; no contributions*

		experience	
Ashritha Dandam	Pharmacy, health informatics.	No SQL or Python experience	Research, support in technology (OLS, K-Means, etc.)
John Burns	Radiology, health informatics. IT Project Manager.	SQL/Python/Project Management	Data analysis and Python lead (managed many analysis methods), data collection, SQL support
Sonali Abhijit Patil	Doctor	Python,R programming and SQL experience.	Research support, some analysis (t-test etc.)
Whitney Walker	Health communications, public health, design, Science, Technology studies, life sciences.	No SQL or Python experience. Some project management experience.	Project management: report proofing and editing; data visualizations, data collection, cleanup, extraction, SQL formatting

2.4 Project Challenges

General Challenges

The biggest overarching challenge was our team's lack of experience with the language and tools. We often felt like we were going into the project blind and had difficulties even putting together a fully robust project proposal when we didn't fully understand topics required, such as machine learning. Often, we had to teach ourselves concepts and tools before they were addressed in class, which was difficult at times and made it nearly impossible to stick entirely to our original timeline for progress. In some cases, we even used tools that were not discussed in class, such as the Plotly library for choropleth maps. However, we were still able to stick to our original plan in most cases, and made appropriate adaptations as our knowledge and experience improved.

Many times the sql database and Jupyter were not responsive, particularly during the first half of the class. There were numerous times that we had planned to complete tasks in phpMyAdmin and Jupyter and were unable to due to lack of responsiveness in the server. This issues lessened as the class went on and became almost nonexistent by the end of the class.

Data Challenges

When we began parsing the data, we experienced some challenges choosing the best factors and outcomes to measure. We relied heavily on public health research to select our final factors and outcomes. When combining those factors, we tried a few different ways of combining the data before settling on a combined view that only included these relevant fields. Initially, we had tried combining the data into a combined table, before realizing a combined view through a joining of the tables would be more effective, so that any updates made to the source tables would be reflected in the view. In general, we had a similar number of data for each field. Unfortunately, the hospital rating contained many null values and had multiple values per FIP and created problems with our combined view. If we did a

standard inner join, our dataset would be greatly reduced to the number of hospital ratings (we had around 3,000 entries for most fields but only around 800 for hospital rating). Instead, we chose to do left joins so that FIP codes that had null or multiple values for hospital rating could be used. Later, we had to remove duplicates for some of our fields to avoid analysis issues. This was easily accounted for with small programming changes, such as using the DISTINCT statement in our SQL queries for our visualizations.

Datasets had various field issues. In many data sets, different nomenclature was used for the same fields (for example, some files spelled out states, others used abbreviations). We overcame this issue by finding reference files that helped us clean up the data and translate all files to a common nomenclature. Figuring out ways to handle null values in some analysis methods such as feature scaling was sometimes difficult.

Additionally, we had originally decided to use ZIP code as a common identifier. In practice, a similar identifier, Federal Information Processing (FIP) code -- a location measure that provides a unique number based upon the combination of county and state -- proved to be a more effective metric. FIP code was required for plotly visualizations and --along with state -- became our primary regional metric. When mapping the data in choropleth map, we originally wanted to show the values for each FIP code. Unfortunately, this was too complex for the Jupyter notebook and Plotly so we had to instead show grouping by state.

We planned on using an income dataset from a U.S. Census report. However, we found that the income data in the countyhealthrankings.org that included some of our other fields such as hospital rating, had more robust income data and switched to that dataset.

Analysis Issues

We promised a lot of analysis in our proposal that we later found out could not be completed and was not appropriate for our data. For example, we switched from logistic to linear regression because our features and outcomes were continuous instead of discrete. Similarly, because the output of linear regression analysis is not binary we are not able to complete a Receiver Operator Characteristic Curve. Even though we were unable to complete some items like these, we were able to complete more analysis than was in the proposal. Now that we have the knowledge learned from the course, our analysis makes sense and uses the most appropriate analysis and visualizations for our data.

Because of our limited knowledge starting with the project, we sometimes pursued analysis methods that ended up not being useful to our overall research question and results. For example, we completed k-means analysis, but later realized this wasn't useful to our research question. We didn't need to do an unsupervised learning method like k-means because we already had labels. We also completed OLS in addition to our linear regression, but realized linear regression was a better way of modeling for our project.

While doing feature scaling we had difficulties dealing with the null values and did not want the nulls to impact our analysis. We handled missing (null) values by imputation, where we replaced the null values with mean of column values.

3. Data Collection

All factor and outcome data used encompasses the entire United States and can be targeted by state and fip/zip code. The hospital quality ratings were collected from an aggregate 2014-2017 Medicare data report; the pollution levels were collected by an aggregate 2010-2014 EPA pollution report. Health outcomes and income statistics were collected from countyhealthrankings.org, which used more than 20 data sources, including the CDC, FBI and US Census.

To ensure all data was connected by common state and FIP code (a unique code that maps state and county) identifiers, two reference files were downloaded. One file showed fip codes for each state, zip code and county and was downloaded from dataworld.com. The other reference file contained a key of different ways states are represented (as initials, fully spelled out etc.), to ensure the state format was the same across datasets and was downloaded from scottontechnology.com.

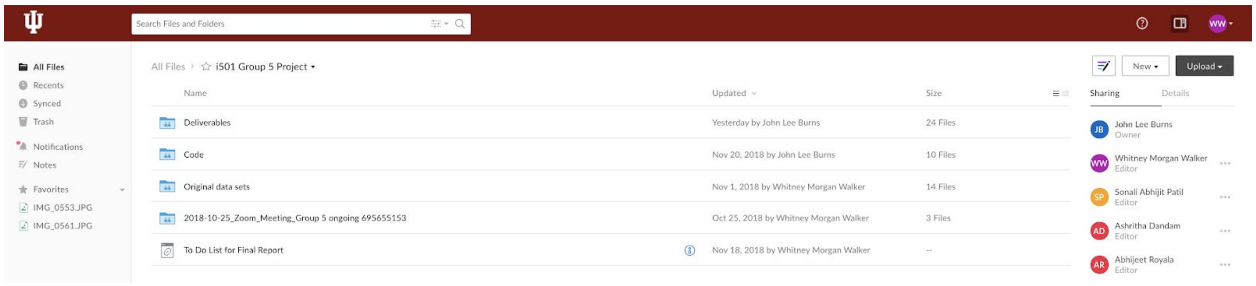
Final Cleaned Up CSVs include:

- Reference Files
    - ZIP_to_FIPS_2010-06.csv
    - US_States_References.csv
- Factor and Outcome Files
    - Hospital _Quality_Ratings_FIP.csv
    - County_Health_Rankings_Trim_FIP.csv
    - Annual_Air_Quality_2016_FIP.csv

4. Data Extraction and Storage

4.1 Data Extraction

Data was downloaded as JSON or CSV files and converted to CSVs for initial review. All original files were stored in a shared group 5 box folder with the other group project materials. We did not remove any attributes from our CSV files and added a common FIP code identifier to all files. We later would include our relevant attributes in the combined SQL view.



Shared IU Box folder with our code, deliverables and other project materials

We chose the following attributes for our combined SQL view because they were most relevant to exploring how select health factors impact select health outcomes. The attributes are shown in their respective categories and given descriptions. Understanding the descriptions is useful for understanding how these attributes relate to the research questions and helpful for understanding the analysis outcomes.

- **Classifier information**
  - fip
    - Also known as a Federal Information Processing Standards code. It uniquely identifies locations based upon a combination of state and county.
  - state
    - All 50 states in the United States
  - county
    - A small level of regional identification within states
  - Population_estimate_value
    - The estimated population for each FIP code
- **Factors**
  - Median_household_income
    - The median income for households in each FIP code
  - Hospital_rating
    - Shows quality ratings on a scale of 1-5 (1, worst; 5 best) for hospitals registered with Medicare.
  - Median_aqi
    - Half of daily AQI values during the year were less than or equal to the median value, and half equaled or exceeded it ("About Air Data Reports", 2018).
  - AQI_ninety_percentile
    - 90 percent of daily AQI values during the year were less than or equal to the 90th percentile value ("About Air Data Reports", 2018).
- **Outcomes**
  - Premature_deaths
    - This metric shows the number of premature deaths that could have been prevented.
  - Low_birthweight
    - Shows current and future morbidity and premature mortality risk for infants born under 2,500 grams. Is a percentage of the population.

4.2 Data Cleaning

As noted in our data extraction section, we chose attributes based upon our research to explore the best factors and outcomes. Instead of removing attributes we didn't want to use in analysis, we chose keep all attributes from our source files. To focus on our chosen attributes, we instead used a phpMyAdmin view, which will be discussed more in the data import section.

ZIP	COUNTYNAME	STATE	STCOUNTYFP
36003	Autauga County	AL	1001
36006	Autauga County	AL	1001
36008	Autauga County	AL	1001
36022	Autauga County	AL	1001
36051	Autauga County	AL	1001
36066	Autauga County	AL	1001
36067	Autauga County	AL	1001
36091	Autauga County	AL	1001
36703	Autauga County	AL	1001
36749	Autauga County	AL	1001
36758	Autauga County	AL	1001
36502	Baldwin County	AL	1003
36507	Baldwin County	AL	1003

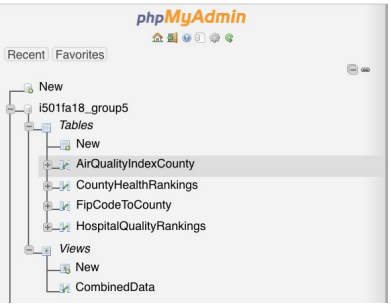
Preview of the reference file that matched FIP code to state, county, and ZIP code

Initially, in our project we had planned on using ZIP codes as the unique identifier and key for each table. However, upon some exploration of our visualization methods, we realized we would need to use FIP codes--a similar geographic notation--instead. Because our original data files didn't include FIP code, we had to find a way to add these codes. Luckily, FIP codes are a unique combination of state and county (two notations already in our

files). Before we could add the FIP codes, we had to ensure all states and counties were formatted in the same way in all files. To do this, we used VLOOKUP and a source data file to standardize county and state formatting. To cross-reference and add accurate FIP codes to our original files, we then created a concatenated unique column of state and county in the reference and source file. We then used this unique column with VLOOKUP in excel to add the FIP codes to all of our files. Once all the codes were added, we removed the concatenated column.

For most of our FIP codes we had one value (and very few null values) with each factor and outcome, which required very little cleaning. The attribute we struggled with though was the the County Hospital Rating. This factor was too significant to our research question to remove from analysis but had many null values for some FIPS and many duplicate values for other FIPS. We did not remove or alter the data directly because we did not want to remove any of the clarity that the original data had. Instead, we accounted for these duplicate and null values in our other visualizations and analysis.

4.3 Data Import



Screenshot of tables and views

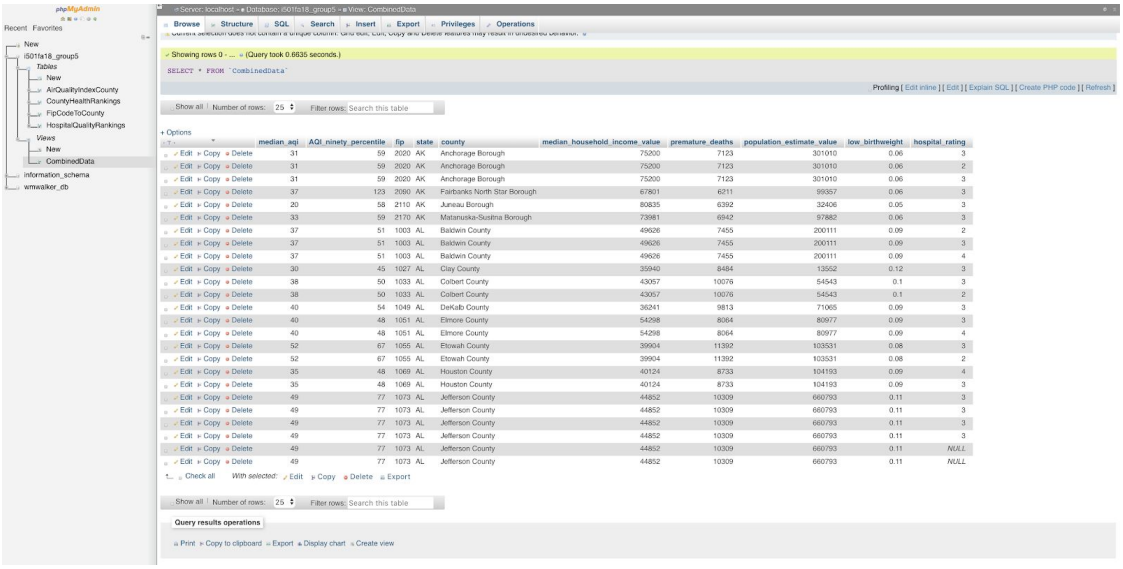
We decided that having a shared database in phpMyAdmin was the best way to store and access our data. Our database, i501fa18_group5, was accessible to all group members and used for all visualizations and analysis. To add all of the data to phpMyAdmin, we converted the cleaned up CSV data files into SQL upload statements using Notepad++ instead of using the phpMyAdmin CSV uploader. This gave us control over columns and how the data was uploaded that the CSV uploader cannot do. We then created a view that combined the relevant fields for our visualizations and analysis.

As noted, we had some null and duplicate values for some of our County Hospital Ratings, making an inner join an ineffective method of joining the tables. When we used inner join, we would lose all of our FIP codes that had null values for County Health Ratings, which reduced our FIP codes from approximately 3,000 entries to approximately 800 entries.

```
CREATE VIEW CombinedData AS
SELECT
AirQualityIndexCounty.median_aqi, AirQualityIndexCounty.AQI_ninety_percentile, CountyHealthRankings.fip, CountyHealthRankings.state, CountyHealthRankings.county,
CountyHealthRankings.median_household_income_value, CountyHealthRankings.premature_deaths, CountyHealthRankings.population_estimate_value,
CountyHealthRankings.low_birthweight, HospitalQualityRankings.hospital_rating
FROM
AirQualityIndexCounty
LEFT JOIN CountyHealthRankings ON AirQualityIndexCounty.fip = CountyHealthRankings.fip
LEFT JOIN HospitalQualityRankings ON AirQualityIndexCounty.fip = HospitalQualityRankings.fip;
```

Screenshot of SQL code used to create CombinedView in phpMyAdmin

Instead, we used left joins, which kept FIP codes that had null or more than one County Hospital Ratings values. Because we then had duplicate rows for some of our FIP codes to account for the sometimes multiple County Hospital Ratings, we had to account for these duplicates in our analysis and visualization code.



Screenshot of CombinedView phpMyAdmin view; note duplicate values to account for multiple hospital ratings

5. Data Analysis

We utilized Python libraries Pandas, Numpy, Pylab, SKLearn, Seaborn, and Matplolib to analyze the data. The 3 independent variables are ‘Hospital Rating’, 'Scaled Median AQI', 'Scaled AQI 90 Percentile', 'Scaled Median Household Income' and the 2 dependent variables are 'Premature Deaths','Low Birthweight.' All variables were continuous. This analysis was separated into checking normality and correlation of variables, and then running linear regression to build a model of the data. All steps below were completed in Jupyter Notebook running Python 3.

5.1 Step 1: Class Creation and Feature Scaling

We created the class FipCodeCombineStats which allows us to store objects and query them. As part of object initialization, the class scales all of the internal variables such that each feature and outcome is between 0 and 1. The outcomes were initially scaled by the population value of the object, however, after testing we determined that scaling outcomes Premature Deaths and Low Birthweight by population was not the correct option for later analysis. The results became too small and were rounded to zero by most functions for Premature Deaths. For Low Birthweight, the numbers were already a percentage of the population and did not need to be scaled. We decided to use the iScale function on the outcomes as well so that these were also scaled between 0 and 1, based upon the minimum and maximum values. We kept all of the factors scaled because it made it easier to analyze and measure the impact for each feature. Our final scaling was built based on the example class from the Guttag Python book Titanic examples (Guttag, 2017). By creating classes we are controlling scaling for each object in an centralized location and because each object feature and outcome has specific variables we are able to recall these easier later. We keep the scaled and non-scaled value in each object so that we can easily switch between scaled/non-scaled values as we do our analysis.

```
#combined stats class maps labels to features by FIP code
class FipCodeCombineStats(object):
    features = ('Median AQI', 'AQI 90 Percentile', 'Median Household Income','Hospital Rating')
    scaledFeatures = ('Scaled Median AQI', 'Scaled AQI 90 Percentile', 'Scaled Median Household Income','Scaled Hospit
al Rating')
    def __init__(self, fip, state, county, median_aqi, AQI_ninety_percentile, median_household_income_value,
                 premature_deaths, population_estimate_value, low_birthweight, hospital_rating):
        self.fip = fip
        self.state = state
        self.county = county

        #outcomes
        self.premature_deaths = premature_deaths
        self.low_birthweight = low_birthweight

        #scaled outcomes by population
        #self.scaled_premature_deaths = premature_deaths/population_estimate_value
        #self.scaled_low_birthweight = low_birthweight/population_estimate_value

        #scaling outcomes by min/max values. This makes features from 0 to 1, making weight easier to compare
        self.scaled_premature_deaths = iScaleFeatures(premature_deaths,3219, 20484)
        self.scaled_low_birthweight = iScaleFeatures(low_birthweight, .039999999, .280000)

        #create the featureVec
        self.featureVec = [median_aqi, AQI_ninety_percentile, median_household_income_value, hospital_rating]

        #scale using iScaleFeatures based on min/max values.  This makes features from 0 to 1, making weight easier to
 compare
        scaled_median_aqi = iScaleFeatures(median_aqi,0,151)
        scaled_AQI_ninety_percentile = iScaleFeatures(AQI_ninety_percentile,4,200)
        scaled_median_household_income_value = iScaleFeatures(median_household_income_value,24868,122641)
        scaled_hospital_rating = iScaleFeatures(hospital_rating,1,5)
        self.scaledFeatureVec = [scaled_median_aqi, scaled_AQI_ninety_percentile, scaled_median_household_income_value
,scaled_hospital_rating]

    def distance(self, other):
        return minkowskiDist(self.scaledFeatureVec,other.scaledFeatureVec) #featureVec or scaledFeatureVec as appropri
ate

    def getLabel(self):
        #change depending on which you are loading
        return self.scaled_premature_deaths
        #return self.scaled_low_birthweight

    def getFeatures(self):
        return self.featureVec

    def getScaledFeatures(self):
        return self.scaledFeatureVec
```

Scaling python code

5.2 Step 2: Creating a Dictionary

We then connected to SQL, query for the data we need from the CombinedData view, iterate through the results, and build a list and dictionary of query results as FipCodeCombineStats objects. We do the query in 2 parts because the hospital rating data has many results per FIP, and we used SQL to average these in one line. The hospital rating data also does not completely join the other data set, so we substitute the average rating for any FIP that does not have one.

The outcome of the functions below are a list of FipCodeCombineStats objects, and a dictionary of FIP to FipCodeCombineStats objects. These are utilized later for either- List: processing through all objects; Dictionary: recalling a specific object by it's FIP code.

```
#get all the data into a dictionary and a list
#Starting the database connection and setting up default values
import mysql.connector
import datetime

mydb = mysql.connector.connect(
  host="localhost",
  user="i501fa18_group5",
  passwd="*85E!4LfLQbQ",
  database="i501fa18_group5"
)

#blank arrays for classes
combinedDict = {}
combinedList = []
hospitalDict = {}

#query to get the hospital_rating and create a hospitalDict.  Fewer FIP codes reported this statistic, so we are runnin
g this seperately to not confuse the other outputs.
cursor = mydb.cursor()
query = ("SELECT avg(hospital_rating), fip FROM CombinedData group by fip HAVING AVG(hospital_rating) >= 0")
cursor.execute(query)
for row in cursor:
    hospitalDict[row[1]] = row[0]

#query for FipCodeToCounty, build class dictionary
query = ("SELECT fip, state, county, median_aqi, AQI_ninety_percentile, median_household_income_value, premature_deaths
, population_estimate_value, low_birthweight FROM CombinedData where premature_deaths is not NULL and low_birthweight i
s not NULL")
cursor.execute(query)
for row in cursor:
    #adding in the hospital rating value
    #appending the average of all other FIP codes to remove NULLS
    hr = float(3.0131)
    if(hospitalDict.get(row[0])):
        hr = float(hospitalDict.get(row[0]))
    currentFip = FipCodeCombineStats(row[0],row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8],hr)
    combinedDict[row[0]] = currentFip
    combinedList.append(currentFip)

#close the database when you are done with it
mydb.close()
```

Creating the dictionary python code

5.3 Step 3: Data Frames

For use with Seaborn and SKLearn libraries analysis, we need to transform the dictionary into a Pandas dataframe object. The easiest method to do this was to load in the objects as columns, then transpose the data frame so that the rows became columns.

```
outDict = {}
count = 0
count2 = 0
#creating a dict of fip lists
for fip in combinedList:
    #first add in the scaled features
    temp = fip.scaledFeatureVec
    #now add the outcomes to the end
    temp.append(fip.scaled_premature_deaths)
    temp.append(fip.scaled_low_birthweight)
    outDict[fip.fip] = temp

#outputting to a pandas dataframe, then rotating it
data_frame = pd.DataFrame(outDict)
data_frame_transposed = data_frame.T
data_frame_transposed.columns = ['Scaled Median AQI', 'Scaled AQI 90 Percentile', 'Scaled Median Household Income','Sc
aled Hospital Rating','Scaled Premature Deaths','Scaled Low Birthweight']
data_frame_transposed
```

Data frames creation python code

The Pandas output looks like the following:

FIP Code	Scaled Median AQI	Scaled AQI 90 Percentile	Scaled Median Household Income	Scaled Hospital Rating	Scaled Premature Deaths	Scaled Low Birthweight
10001	0.258278	0.290816	0.300727	0.500000	0.264292	0.166667
10003	0.291391	0.357143	0.406697	0.750000	0.232262	0.208333
10005	0.271523	0.290816	0.290960	0.666675	0.225717	0.166667
1003	0.245033	0.239796	0.253219	0.500000	0.245352	0.208333
1027	0.198675	0.209184	0.113242	0.500000	0.304952	0.333333
1033	0.251656	0.234694	0.186033	0.375000	0.397162	0.250000
1049	0.264901	0.255102	0.116320	0.500000	0.381929	0.208333
1051	0.264901	0.224490	0.301003	0.625000	0.280626	0.208333

5.4 Step 4: Comparing Normality

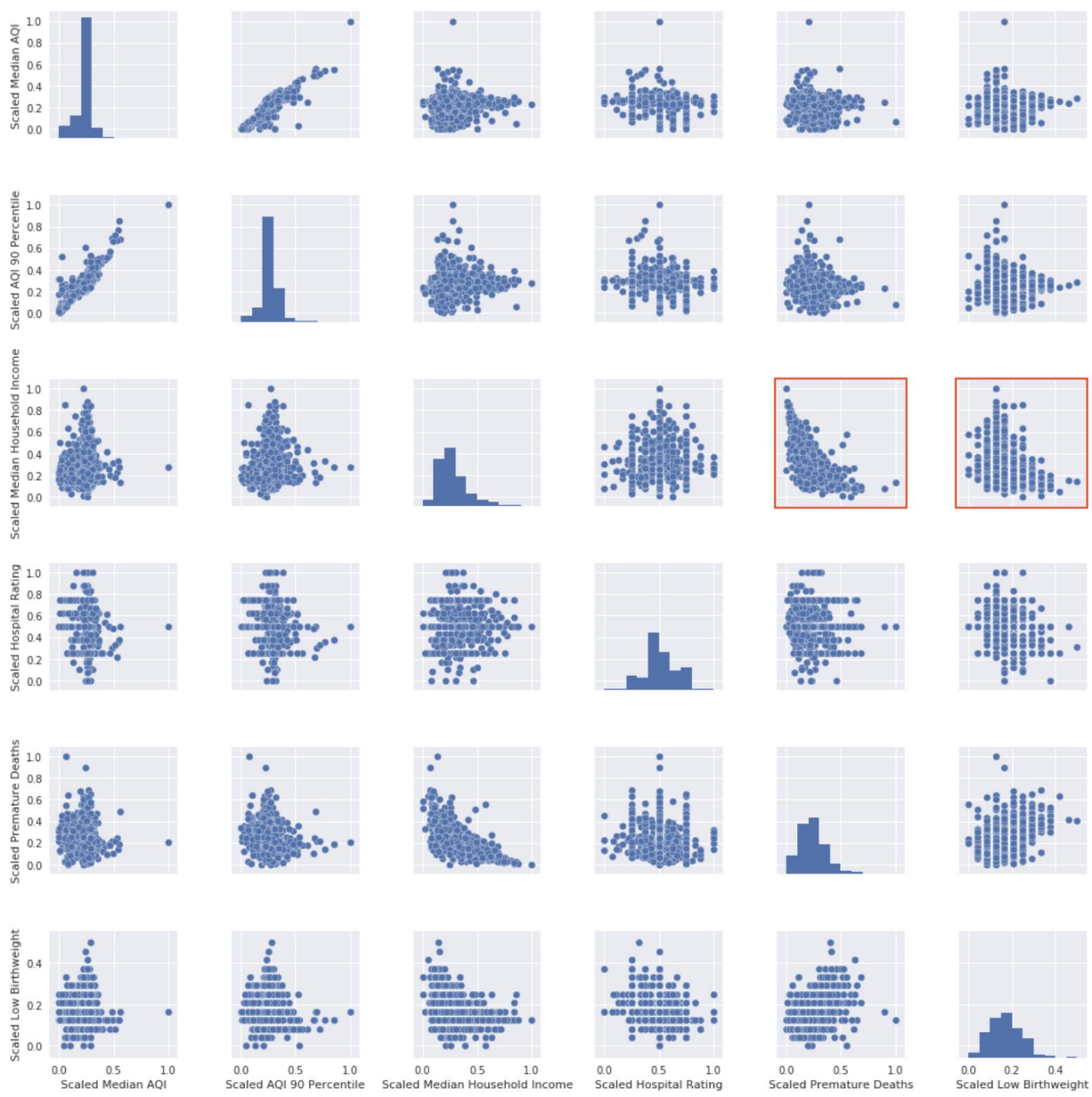
First, we checked for normal distribution of our factors and outcomes in the data. Knowing if our data is normally distributed was helpful to ensure we used the proper models for analysis. We tested for normality by using the stats.normal test and scaled values.

```
#Normal Test
for i in ['Scaled Median AQI', 'Scaled AQI 90 Percentile', 'Scaled Median Household Income','Scaled Hospital Rating','S
    print('Distribution of', i, ': ',stats.normaltest(data_frame_transposed[i]))

for i in ['Scaled Median AQI', 'Scaled AQI 90 Percentile', 'Scaled Median Household Income','Scaled Hospital Rating','S
    print(i, '\nMean of: ',data_frame_transposed[i].mean())
    print('Standard Deviation of: ',data_frame_transposed[i].std())
```

The null hypothesis for stats.normal test is that the data is normally distributed. Because our p values were all less than 0.05, we were able to confidently reject the null that the data is a normal distribution. Therefore, our data is not normally distributed.

```
Distribution of Scaled Median AQI : NormaltestResult(statistic=230.335140113855, pvalue=9.624093421549685e-51)
Distribution of Scaled AQI 90 Percentile : NormaltestResult(statistic=433.4585765346251, pvalue=7.51045137467058e-95)
Distribution of Scaled Median Household Income : NormaltestResult(statistic=260.8633764038711, pvalue=2.2606719649741
085e-57)
Distribution of Scaled Hospital Rating : NormaltestResult(statistic=16.682487089402716, pvalue=0.0002384756033977546
2)
Distribution of Scaled Premature Deaths : NormaltestResult(statistic=196.4616635906996, pvalue=2.182186167416937e-43)
Distribution of Scaled Low Birthweight : NormaltestResult(statistic=106.50047615762648, pvalue=7.476794467766934e-24)
```

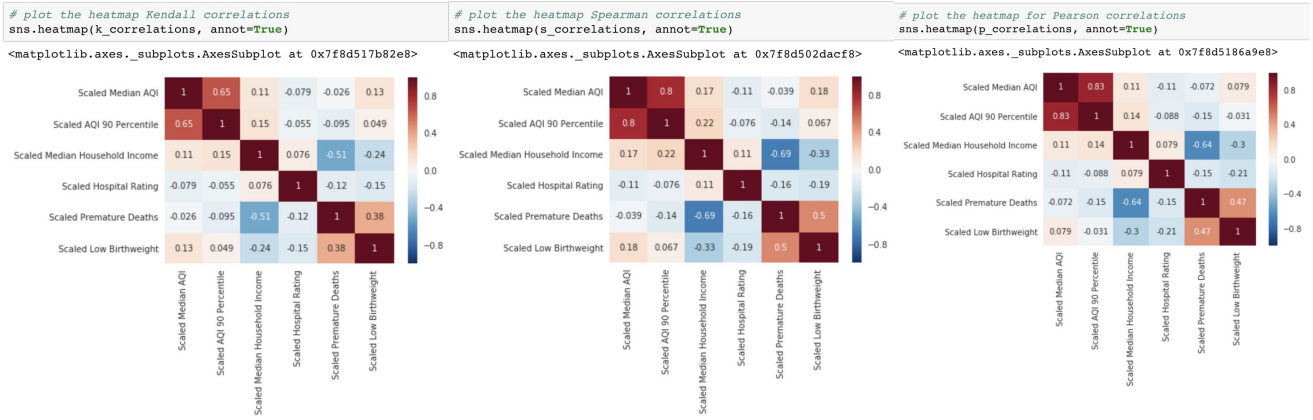
Then we compared normality of the variables using histograms and created scatter plots of each variable compared to the others by FIP code using the Seaborn pairplot function. Each dot in the scatter plots is a different FIP code showing the relationship between the Y left side variable and X bottom side variable. This allows us to perform some sanity checks, such as Median AQI being linear to 90 Percentile AQI. This shows that much of our data is not perfectly normally distributed but does fit in the normal family of distributions. It also shows a clear correlation between Median Household Income and the 2 outcomes (highlighted in red below). 90 Percentile AQI and Median AQI may have correlation with either outcome, but it is very hard to tell from these plots.



5.5 Step 5: Testing Correlation

From there we can calculate correlations using Pandas corr function. There are 3 types of correlations corr can calculate – Pearson, Kendall, and Spearman. Pearson is the most commonly used correlation map and is used to measure the relationship between variables with normal distribution. Kendall measures the strength of dependence between two variables and Spearman measures the degree of association between two variables that are non-parametric, which means these models are primarily for non-normally distributed data ("Correlation (Pearson, Kendall, Spearman)"). With these models, we not only wanted to determine the relationships between factors and outcomes, we also wanted to see the relationship amongst factors. For example, by comparing media AQI and 90th percentile AQI, we were able to

perform a check to see if the tests were performing as planned--that is, we expected these two factors to be strongly correlated because they were similar measures and they were as seen in the results. If we were to continue to pursue this research topic, we'd use these correlations to remove weakly correlated factors from our combined model.



Each heat map tells a similar story of the numbers, with slightly different final calculations. The darker the number the stronger the correlation. For example, the Median AQI and 90 Percentile AQI are clearly very related as expected, because they are different ways of looking at similar data. The features and outcomes that are most related are the Median Household Income to Premature Deaths (as income goes up, premature deaths go down), then Median Household Income to Low Birthweight (as income goes up, low birthweights go down), followed by Hospital Rating to Low Birthweight (as hospital ratings go up, low birthweights go down). We expected to see a negative relationship between Median Household Income and our outcomes, as this would indicate that those with more money can afford better healthcare. Of Median AQI and 90 Percentile AQI, only the relationship between AQI 90 percentile and premature deaths was significant enough to warrant further investigation; it is still a very low correlation.

5.6 Step 6: Linear Regression Analysis

Now that we know which variables correlate, we can perform linear regression analysis and see if this matches the expected correlations. This consists of separating our Pandas data frame in to individual columns, then for each training set separating the training/testing data with a 60/40 split. This allows us to train the linear regression model with 60% of the available data (1024 rows) and test against the 40% remaining. Once the data was split we created a sklearn LinearRegression object, and fit the training data to it. Once the model was generated we used it to predict outcomes with the test feature data, and then created a scatter plot of generated outcomes vs expected outcomes. Because we know that only 3 feature/outcome pairs have significant correlations, we could just run linear regression against these. However, because it was easy to complete, we went ahead and ran linear regression against all 8 feature/outcome pairs. Shown below is code to generate the first 2 linear regression models and plot, the remaining 6 are similar but with different variables.

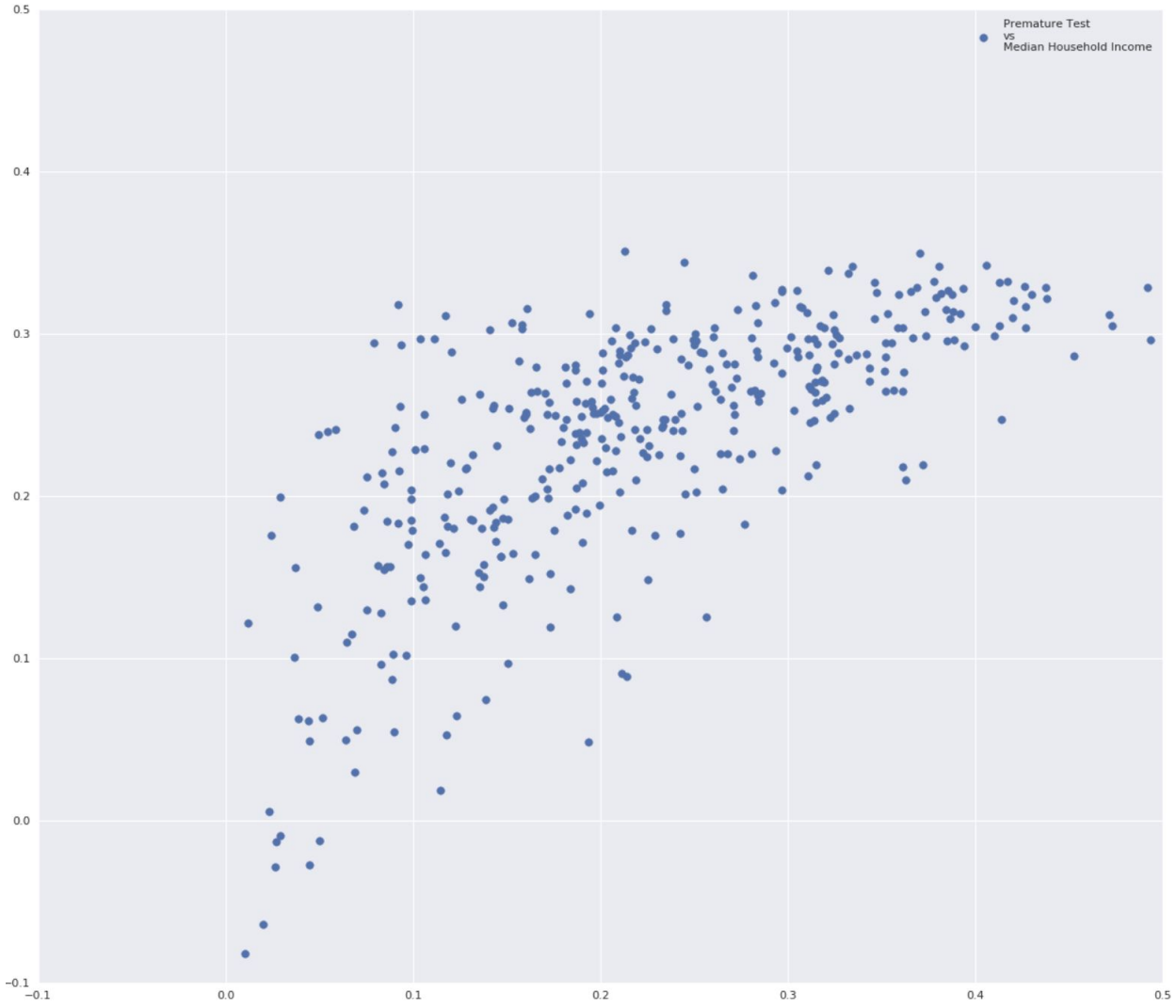
```
#Building testing lists
x_sma = data_frame_transposed[['Scaled Median AQI']]
x_sa90p = data_frame_transposed[['Scaled AQI 90 Percentile']]
x_smhi = data_frame_transposed[['Scaled Median Household Income']]
x_hr = data_frame_transposed[['Scaled Hospital Rating']]
x_combined = data_frame_transposed[['Scaled Median AQI','Scaled AQI 90 Percentile','Scaled Median Household Income','Sca
Y1 = data_frame_transposed[['Scaled Premature Deaths']]
Y2 = data_frame_transposed[['Scaled Low Birthweight']]

#set plot sizes to something more reasonable
plt.figure(figsize=(15, 15))

#building and test data
#Scaled Median AQI vs Scaled Premature Deaths
X_train_smap, X_test_smap, y_train_smap, y_test_smap = sklearn.model_selection.train_test_split(
    x_sma, Y1, test_size=0.4, random_state=101)
lm_smap = LinearRegression()
lm_smap.fit(X_train_smap,y_train_smap)
predictions_smap = lm_smap.predict(X_test_smap)
plt.subplot(5, 2, 1)
plt.scatter(y_test_smap, predictions_smap, label='Premature Death Test\nvs\nMedian AQI')
plt.legend(loc='best')

#Scaled Median AQI vs Low Birthweight
X_train_smab, X_test_smab, y_train_smab, y_test_smab = sklearn.model_selection.train_test_split(
    x_sma, Y2, test_size=0.4, random_state=101)
lm_smab = LinearRegression()
lm_smab.fit(X_train_smab,y_train_smab)
predictions_smab = lm_smab.predict(X_test_smab)
plt.subplot(5, 2, 2)
plt.scatter(y_test_smab, predictions_smab, label='Low Birthweight Test\nvs\nMedian AQI')
plt.legend(loc='best')
```

We also chose to generate a combined feature set model to see how this affected predictions. Unlike the other models which used 1 feature variable to predict 1 outcome variable, this uses 4 feature variables to predict 1 outcome variable each.

The plots below show the Outcome Test data (known values) on the X axis with the Model prediction values on the Y axis:

```
X values are known, Y values are predicted
```

As expected based on the correlations, Median AQI and 90 Percentile AQI are not able to predict either Premature Deaths or Low Birth weights reliably. Median Household Income (red star) is so influential in our model that the combined product is essentially the same except for minor variations on some outliers. Because the Low Birthweight outcome is so small in the original data, it is very hard to visualize the outcome of the model. It appears to do okay but is not as correlated as the Premature Deaths to Median Household Income. Let's take a closer look at Premature Deaths to Median Household Income by enlarging the graph and removing some of the outliers.

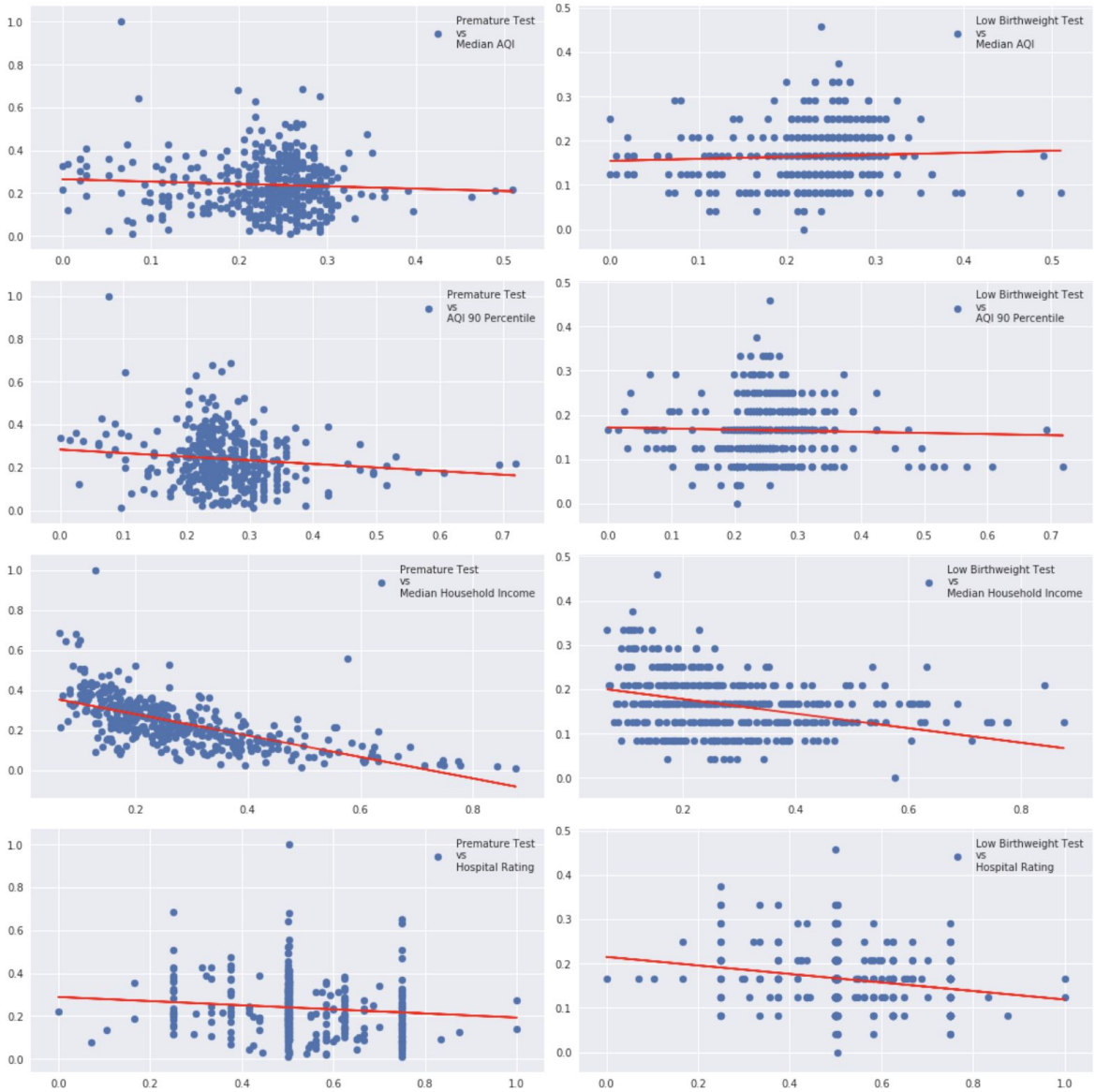


This looks like a pretty good match for our data. Note – if it were perfect, this would be a diagonal line.

5.7 Step 7: Outcome Test Vs. Outcome Model Predictions

Instead of viewing this as Outcome Test vs Outcome Model Predictions, we then created a scatter plot of the Test Feature as X values and the Test Outcome as Y values and then we will overlay a line of Test Feature X values and Model Predicted Outcomes as Y values. The code is very similar, with a few modifications. This visualization allows us to see how our models inputs correlate to outputs and more easily compare the test data set (scatter) to the models' fitting (line).

```
#set plot sizes to something more reasonable
plt.figure(figsize=(15, 15))

#plotting test data
#Scaled Median AQI vs Scaled Premature Deaths
plt.subplot(4, 2, 1)
plt.scatter(X_test_smap, y_test_smap, label='Premature Test\nvs\nMedian AQI')
plt.plot(X_test_smap, predictions_smap, color='red')
plt.legend(loc='best')

#Scaled Median AQI vs Low Birthweight
plt.subplot(4, 2, 2)
plt.scatter(X_test_smab, y_test_smab, label='Low Birthweight Test\nvs\nMedian AQI')
plt.plot(X_test_smab, predictions_smab, color='red')
plt.legend(loc='best')
```



The output plots are very similar to the correlations we calculated above. Only 1 graph has a positive slope – Low Birthweight vs Median AQI – with the remaining having negative correlations. Again, the Median Household Income seems to fit both outcomes the best. Note that the combined dataset is not plotted here, it would require a 4-dimensional graph to correctly plot the 3 features to one outcome.

5.8 Step 8: Find the R^2, MSE, and RMSE values

We use the SKLearn Metrics package to create R^2 and Mean Squared Error (MSE) values for each prediction, and then take the square root of the MSE to get RMSE values.

```
print('Premature Test vs Median AQI\nR^2 value: ',r2_score(y_test_smap,predictions_smap),'\nMean Squared Error: ',mean_square
d_error(y_test_smap,predictions_smap),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_smap,predictions_smap)),'\n')
print('Low Birthweight Test vs Median AQI\nR^2 value: ',r2_score(y_test_smab,predictions_smab),'\nMean Squared Error: ',mean_
squared_error(y_test_smab,predictions_smab),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_smab,predictions_smab)),'\n')
print('\nPremature Test vs AQI 90 Percentile\nR^2 value: ',r2_score(y_test_sa90pp,predictions_sa90pp),'\nMean Squared Error:
',mean_squared_error(y_test_sa90pp,predictions_sa90pp),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_sa90pp,predictions_sa90p
p)),'\n')
print('Low Birthweight Test vs AQI 90 Percentile\nR^2 value: ',r2_score(y_test_sa90pb,predictions_sa90pb),'\nMean Squared Err
or: ',mean_squared_error(y_test_sa90pb,predictions_sa90pb),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_sa90pb,predictions_s
a90pb)),'\n')
print('\nPremature Test vs Median Household Income\nR^2 value: ',r2_score(y_test_smhip,predictions_smhip),'\nMean Squared Err
or: ',mean_squared_error(y_test_smhip,predictions_smhip),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_smhip,predictions_smhi
p)),'\n')
print('Low Birthweight Test vs Median Household Income\nR^2 value: ',r2_score(y_test_smhib,predictions_smhib),'\nMean Squared
Error: ',mean_squared_error(y_test_smhib,predictions_smhib),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_smhib,predictions_s
mhib)),'\n')
print('\nPremature Test vs Hospital Rating\nR^2 value: ',r2_score(y_test_hrp,predictions_hrp),'\nMean Squared Error: ',mean_s
quared_error(y_test_hrp,predictions_hrp),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_hrp,predictions_hrp)),'\n')
print('Low Birthweight Test vs Hospital Rating\nR^2 value: ',r2_score(y_test_hrb,predictions_hrb),'\nMean Squared Error: ',me
an_squared_error(y_test_hrb,predictions_hrb),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_hrb,predictions_hrb)),'\n')
print('\nPremature Test vs Combined Predict\nR^2 value: ',r2_score(y_test_combinedp,predictions_combinedp),'\nMean Squared Er
ror: ',mean_squared_error(y_test_combinedp,predictions_combinedp),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_combinedp,pre
dictions_combinedp)),'\n')
print('Low Birthweight Test vs Combined Predict\nR^2 value: ',r2_score(y_test_combinedb,predictions_combinedb),'\nMean Square
d Error: ',mean_squared_error(y_test_combinedb,predictions_combinedb),'\nRoot MSE: ',sqrt(mean_squared_error(y_test_combinedb
,predictions_combinedb)),'\n')
```

```
Premature Test vs Median AQI
R^2 value:  0.0028954365725587516
Mean Squared Error:  0.015588857424270975
Root MSE:  0.12485534599796268

Low Birthweight Test vs Median AQI
R^2 value:  0.010576813080859782
Mean Squared Error:  0.004001199474535355
Root MSE:  0.06325503517140241


Premature Test vs AQI 90 Percentile
R^2 value:  0.023672485613066785
Mean Squared Error:  0.01526402644157421
Root MSE:  0.12354766870149436

Low Birthweight Test vs AQI 90 Percentile
R^2 value:  0.0007015300190894 58
Mean Squared Error:  0.004041134840837694
Root MSE:  0.0635699208811659


Premature Test vs Median Household Income
R^2 value:  0.4120935298083195
Mean Squared Error:  0.009191403267799244
Root MSE:  0.09587180642816345

Low Birthweight Test vs Median Household Income
R^2 value:  0.06906433241167009
Mean Squared Error:  0.0037646775952149343
Root MSE:  0.06135696859538397


Premature Test vs Hospital Rating
R^2 value:  0.026949510798084497
Mean Squared Error:  0.015212793020066853
Root MSE:  0.12334015169468072

Low Birthweight Test vs Hospital Rating
R^2 value:  0.04691396830278993
Mean Squared Error:  0.0038542530432183184
Root MSE:  0.06208263076914765


Premature Test vs Combined Predict
R^2 value:  0.43437610881083044
Mean Squared Error:  0.008843034641422503
Root MSE:  0.0940374108609042

Low Birthweight Test vs Combined Predict
R^2 value:  0.1421299379918891
Mean Squared Error:  0.003469202346080641
Root MSE:  0.05889993502611562
```

We can tell from these results that the Median Household Income vs Premature Deaths outcome models the best fit out of any solo input, because it is the closest to an R^2 value of 1. The Combined model is able to fit the Premature Deaths outcome slightly better and this is reflected in the graphical representation as well. Interestingly, the combined model is able to predict the Low Birthweight outcome significantly better than other models, although it's R^2 value is only .142. Because our features and Outcomes are all scaled between 0 and 1 we are able to easily evaluate the RMSE. For the 3 models described the RMSE is .09 or below. This means that the models have a error or approximately 9% RMSE which is pretty good for the data set we are looking at.

5.9 Step 9: Monte Carlo Linear Regression

Finally, we perform Monte Carlo linear regression trials to create a more accurate model. This is based on the book "Introduction to Computation and Programming Using Python, 2e" by Guttag Titanic Example for logistic regression (Guttag, 2017).

Note: We had originally planned on completing a logistic regression model in our proposal, however, we found that logistic regression was only applicable to discrete or bin-able outcomes.

Because our features and outcomes are continuous logistic regression is not applicable.

```
#linear regression running below
stats,premature_weights, low_weights = [],[[], [], [], []],[[], [], [], []] #3 total weights possible
premature_MSE = 0
low_MSE = 0
numTrials = 200
for i in range(numTrials):
    training, testSet = divide80_20(combinedList) #divide the training and test data sets
    featureVecs = [[],[],[],[]]
    labels = [[],[]]
    predictionData = [[],[],[],[],[],[]]
    xVals = [[],[],[]]
    for fip in training:
        featureVecs[0].append(fip.scaledFeatureVec)
        featureVecs[1].append(fip.scaledFeatureVec[0])
        featureVecs[2].append(fip.scaledFeatureVec[1])
        featureVecs[3].append(fip.scaledFeatureVec[2])
        labels[0].append(fip.scaled_premature_deaths)
        labels[1].append(fip.scaled_low_birthweight)

    for fip in testSet:
        predictionData[0].append(fip.scaledFeatureVec)
        predictionData[1].append(fip.scaledFeatureVec[0])
        predictionData[2].append(fip.scaledFeatureVec[1])
        predictionData[3].append(fip.scaledFeatureVec[2])
        predictionData[4].append(fip.scaled_premature_deaths)
        predictionData[5].append(fip.scaled_low_birthweight)

    #find weights for premature deaths dataset
    regr_premature = sklearn.linear_model.LinearRegression()
    regr_premature.fit(featureVecs[0],labels[0])
    for i in range(len(FipCodeCombineStats.features)):  #switch features or scaledFeatures not really necessary as they
are the same length
        #print('Feature weight of: ' + FipCodeCombineStats.features[i] + ' ' + str(regr.coef_[i]))
        premature_weights[i].append(regr_premature.coef_[i])

    #find weights for low birthweight dataset
    regr_low = sklearn.linear_model.LinearRegression()
    regr_low.fit(featureVecs[0],labels[1])
    for i in range(len(FipCodeCombineStats.features)):  #switch features or scaledFeatures not really necessary as they
are the same length
        #print('Feature weight of: ' + FipCodeCombineStats.features[i] + ' ' + str(regr_low.coef_[i]))
        low_weights[i].append(regr_low.coef_[i])

    # Make predictions using the testing sets
    premature_predictions = regr_premature.predict(predictionData[0])
    low_predictions = regr_low.predict(predictionData[0])
    premature_MSE += mean_squared_error(predictionData[1], premature_predictions)
    low_MSE += mean_squared_error(predictionData[1], low_predictions)

print('\nAverages for', numTrials, 'trials')
print('Premature Deaths: ')
for feature in range(len(premature_weights)):
    featureMean = sum(premature_weights[feature])/numTrials
    featureStd = stdDev(premature_weights[feature])
    print(' Mean weight of', FipCodeCombineStats.features[feature],
          '=', str(round(featureMean, 3)) + ',',
          '95% confidence interval =', round(1.96*featureStd, 3))
print(' Average Mean Squared Error over trials: ' + str(premature_MSE/numTrials))

print('\nLow Birth Weights: ')
for feature in range(len(low_weights)):
    featureMean = sum(low_weights[feature])/numTrials
    featureStd = stdDev(low_weights[feature])
    print(' Mean weight of', FipCodeCombineStats.features[feature],
          '=', str(round(featureMean, 8)) + ',',
          '95% confidence interval =', round(1.96*featureStd, 8))
print(' Average Mean Squared Error over trials: ' + str(low_MSE/numTrials))
```

Essentially, this code loops through a number of trials, creating 80/20 train/test data sets, then creates combined linear regression models for both outcomes, and storing the weights in an output list of lists.

```
Averages for 200 trials
Premature Deaths:
 Mean weight of Median AQI = 0.169, 95% confidence interval = 0.047
 Mean weight of AQI 90 Percentile = -0.262, 95% confidence interval = 0.032
 Mean weight of Median Household Income = -0.52, 95% confidence interval = 0.01
 Mean weight of Hospital Rating = -0.061, 95% confidence interval = 0.011
 Average Mean Squared Error over trials: 0.02080121166682837

Low Birth Weights:
 Mean weight of Median AQI = 0.33183255, 95% confidence interval = 0.03469919
 Mean weight of AQI 90 Percentile = -0.27552073, 95% confidence interval = 0.02419677
 Mean weight of Median Household Income = -0.13510813, 95% confidence interval = 0.0068125
 Mean weight of Hospital Rating = -0.08999026, 95% confidence interval = 0.00703325
 Average Mean Squared Error over trials: 0.019254805096461065
```

These weights are used to generate the slope of the line for our combined model and provide a sense of how much each feature affects the outcome. The equation could look like (.169 * MAQI) + (-.262 * AQI90) + (-.52 * MHI) + (-.061 * HR) = Premature Deaths. Entering the known FIP code values for each feature would output the models predicted value. The 95% confidence interval values can be used as +/- the value for the feature coefficient, this expands the model to predict a wider range of values.

Our average MSE is not optimal, if we take the square root to get RSME we get Premature Deaths RMSE of .1442 and Low Birth Weights RMSE of .1387. Because our features are scaled between 0 and 1 these values correlate to about 14%, which is performing worse than the single run linear regression performed in step 8. This is likely due to overfitting of the model with 80/20 instead of 60/40 train/test data.

6. Data Visualization

We created data visualizations for each factor and outcome by U.S. state in Plotly, a python library. We chose the state level because it provided an easily-understandable and comparable view for users. Additionally, any more granular regional views (FIP code etc) were too taxing on the Jupyter notebook and had difficulties loading. We connected the Plotly library with our SQL data using the Jupyter notebook and code as shown in an example below. We created unique SQL scripts to pull the data for each visualization, paying particular care to the way the data was pulled and grouped. For example, we didn't want to represent duplicate values as created in our SQL combined view, so we used distinct values.

```
##Outcome: Premature Deaths

#SQL connection imports
import MySQLdb

#plotly imports
import plotly.plotly as py
import plotly.figure_factory as ff

#general data imports
import numpy as np
import pandas as pd

#connection
conn = MySQLdb.connect(host="localhost", user="i501fa18_group5", passwd="*85E!4LfLQbQ", db="i501fa18_group5")
cursor = conn.cursor()
sql = "SELECT avg(distinct(premature_deaths)/population_estimate_value)*100, state FROM CombinedData group by state"
df = pd.read_sql(sql, con=conn)
```

Connection to SQL database view for plotly visualization

6.1 Python Choropleth State Maps

We used data frames for the visualizations and followed a Plotly tutorial to ensure the data was appropriately represented. We chose to keep the visualizations separate to keep the visualizations

cleaner and easier to view. With the choropleth maps, users can quickly see the level that each factor and outcome are present in each state. We chose a scale with a gradient of 6 colors so that we would show nuance in the different ranges the values could be. In addition to showing the data statically, users can interactively peruse the data by hovering over each state to dig deeper into each of the values. We had a total of 6 visualizations, one for each factor and outcome. A purple color scheme was used for the factors, because it is a neutral color, not representing negative or positive correlations. The outcomes--because they were all negative (bad) outcomes, had a red color scheme.

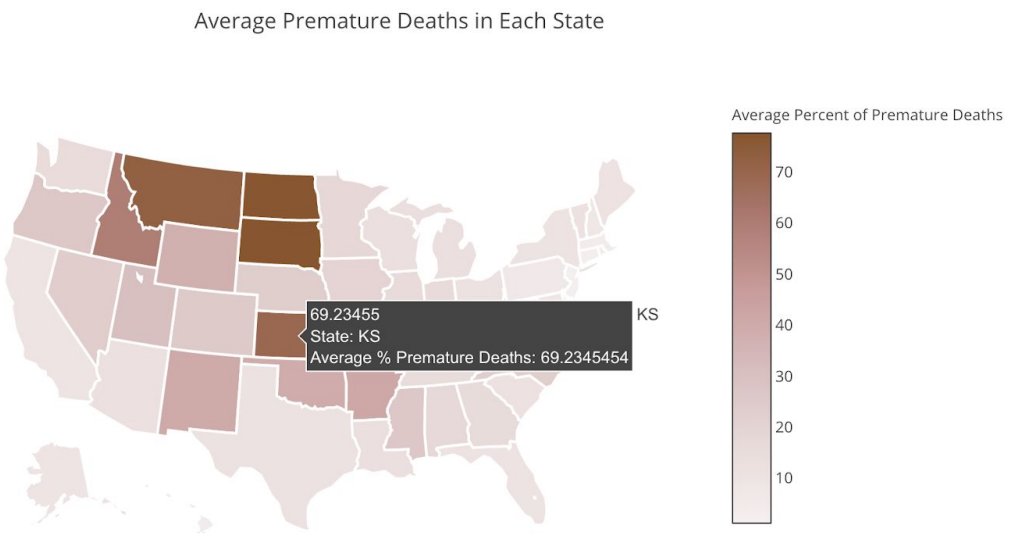
```
for col in df.columns:
    df[col] = df[col].astype(str)

scl = [[0.0, 'rgb(247,240,240)'],[0.2, 'rgb(235,218,218)'],[0.4, 'rgb(220,188,189)'],
            [0.6, 'rgb(208,154,154)'],[0.8, 'rgb(177,117,107)'],[1.0, 'rgb(143,84,39)']]

df['text'] = df['state'] + '<br>' + 'Average % of Low Birthweights: '+df['avg(distinct(low_birthweight))*100']

data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = df['state'],
        z = df['avg(distinct(low_birthweight))*100'].astype(float),
        locationmode = 'USA-states',
        text = df['text'],
        marker = dict(
            line = dict (
                color = 'rgb(255,255,255)',
                width = 2
            ) ),
        colorbar = dict(
            title = "Low Birthweights (%)")
        ) ]

layout = dict(
        title = 'Average Percent of Low Birthweights by State',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)'),
             )

fig = dict( data=data, layout=layout )
py.iplot( fig, filename='d3-choropleth-map' )
```

Sample code for Plotly visualization with low birthweight

Outcome Visualizations

The states with the highest incidents of premature deaths were:

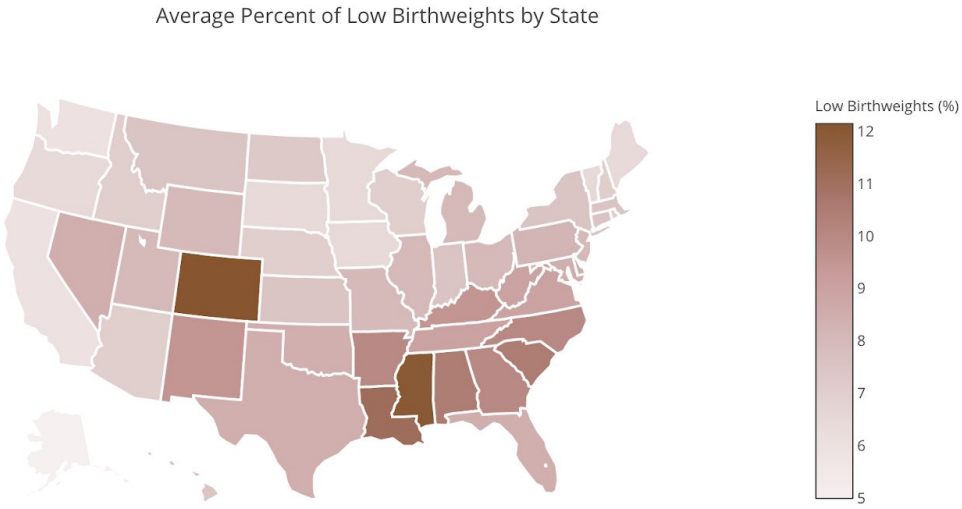
- Montana
- North Dakota
- South Dakota



Visualization for premature deaths outcome; note the red color scheme that represents negative outcomes and the grey box that shows up when the user hovers over the visualization

The states with the highest incidents of low birth weights were:

- Louisiana
- Mississippi
- Colorado

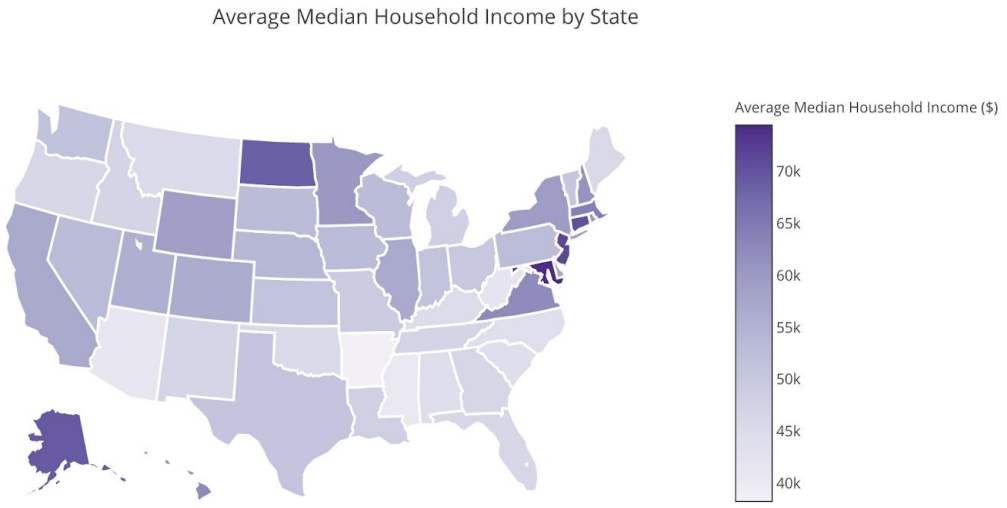


Visualization for an outcome; note the red color scheme that represents negative outcomes

Factor Visualizations

The states with the highest median household income are:

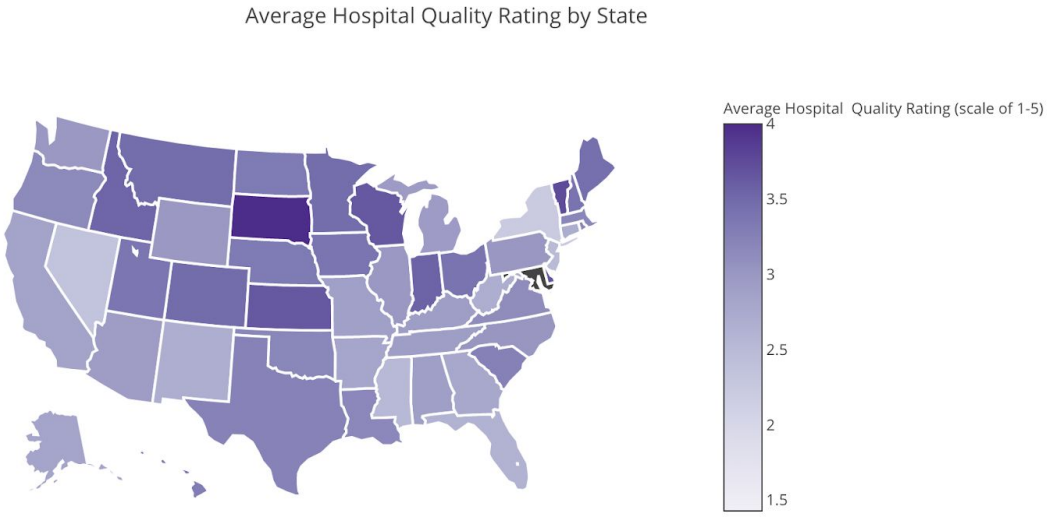
- Maine
- New Jersey
- Connecticut



Visualization for a factor; note the purple color scheme that represents neural factor

The states with the lowest median hospital quality rating are:

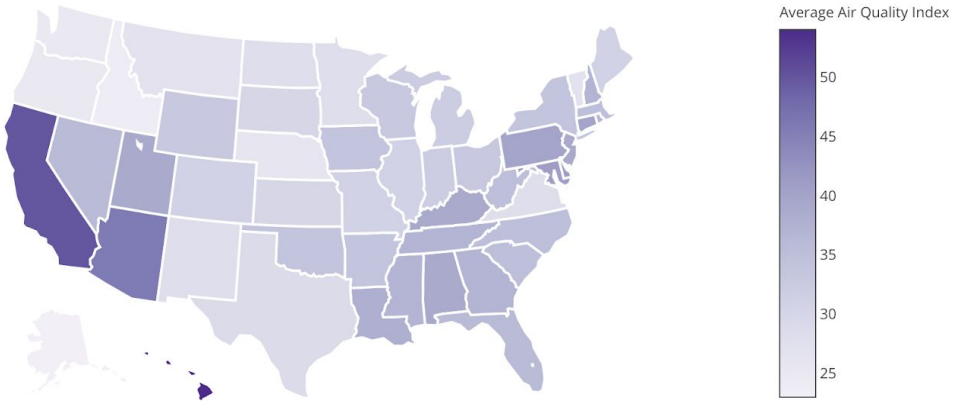
- Nevada
- New York
- New Jersey



The states with the highest average median air quality index are:

- California
- Nevada
- Hawaii

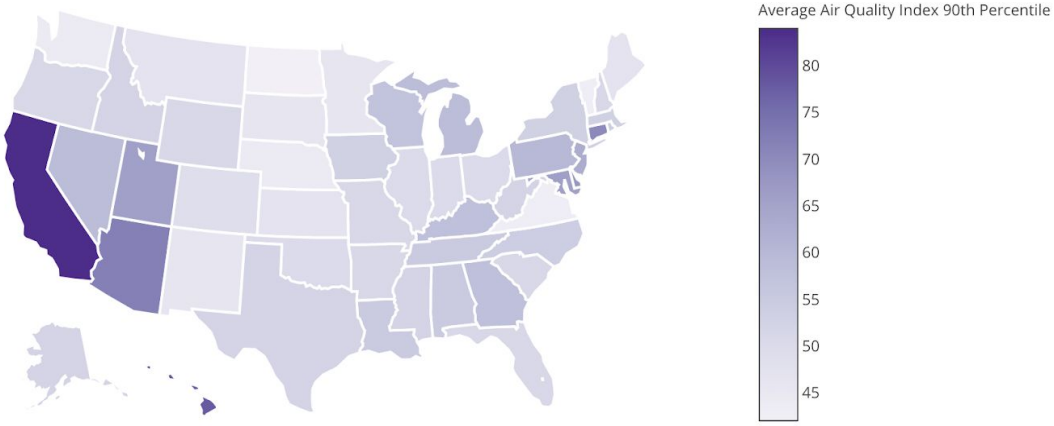
Average Median Air Quality Index by State



The states with the highest average air quality index 90th percentile are:

- California
- Nevada
- Hawaii

Average Air Quality Index 90th Percentile by State



7. Summary of Findings

Results

The team explored to see if the data would show significant relationships between health outcomes and the chosen health factor variables for each region so that policy makers and community members can have a positive impact on their community's health outcomes by targeting specific factors.

From our research and analysis we were able to test our hypotheses. For our first hypothesis, the choropleth visualizations confirm that health factors and outcomes do differ by region in the U.S.

- The states that have the worst air quality indices (AQI) and could use the most intervention in terms of air quality as seen in the average median AQI index and average median AQI 90th percentile measures, are:
    - Hawaii
    - California
    - Nevada
- The state that have the lowest hospital quality ratings and could use intervention are:
    - Nevada
    - New York
    - New Jersey
- The states with the highest median household income are:
    - Maine
    - New Jersey
    - Connecticut
- The states with the highest incidence of low birth weights are:
    - Louisiana
    - Mississippi
    - Colorado
- The states with the highest incidence of premature deaths were:
    - Montana
    - North Dakota
    - South Dakota

Therefore, lawmakers can use this information to benchmark their states and target specific areas of weaknesses in public health. The second hypothesis results will help lawmakers understand which factors are connected to outcomes and are worth the time and effort required for intervention.

Our second hypothesis is not clearly confirmed or rejected. However, we did notice trends. If we break the hypothesis into 3 hypothesis we get the following:

- Datasets will show that by region:
    - High income positively correlates with positive health outcomes - Confirmed
    - High hospital quality ratings positively correlate with positive health outcomes - Not Confirmed
    - High pollution rates positively correlate with negative health outcomes - Not Confirmed

1 of 3 hypothesis are definitively confirmed using our data. That is, high income correlates with positive health outcomes. The other 2 hypothesis are not fully supported by our data set.

8. Limitations

The primary limitation is our collective knowledge and expertise in data science. While we feel comfortable with our analysis and findings, we think with a deeper understanding and more time, our project and analysis could be potentially improved. For example, with a better understanding, we could have thrown out some of our unrelated factors (such as median AQI and premature death as seen in our pearson testing) from testing and found more strongly related factors to test with our combined model

Our datasets are somewhat limited. To more thoroughly examine how our factors and outcomes are related, we would like to have more longitudinal data over time from more sources. We do not fully know how each and every piece of data was acquired so any bias in the datasets we used could be minimized by using more datasets from more sources. Additionally, some of our factors would benefit simply from more data. For example, we didn't have a county hospital rating for each FIP code.

9. Appendix

- Link to full SQL database script
- Link to full analysis and visualization python code
- Link to full Visualization python code
- All final deliverables

```
import numpy as np
import pylab
import sklearn
import sklearn.linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import preprocessing
from sklearn import utils
import random
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
```

Classes imported

```
#defining functions
#page 247 Figure 15.8 Variance and standard deviation
def variance(X):
    """Assumes that X is a list of numbers.
       Returns the standard deviation of X"""
    mean = sum(X)/len(X)
    tot = 0.0
    for x in X:
        tot += (x - mean)**2
    return tot/len(X)

def stdDev(X):
    """Assumes that X is a list of numbers.
       Returns the standard deviation of X"""
    return variance(X)**0.5

#page 379 Figure 22.6 Minkowski distance
def minkowskiDist(v1, v2, p):
    """Assumes v1 and v2 are equal-length arrays of numbers
       Returns Minkowski distance of order p between v1 and v2"""
    dist = 0.0
    for i in range(len(v1)):
        dist += abs(v1[i] - v2[i])**p
    return dist**(1/p)

#page 400 Figure 23.14 Scaling Attributes
def iScaleFeatures(vals,minVal,maxVal):
    """Assumes vals is a sequence of floats"""
    #minVal, maxVal = min(vals), max(vals)
    fit = pylab.polyfit([minVal, maxVal], [0, 1], 1)
    #print(pylab.polyval(fit, vals))
    return pylab.polyval(fit, vals)

#page 407 of book Figure 24.4 Functions for evaluating classifiers
def accuracy(truePos, falsePos, trueNeg, falseNeg):
    numerator = truePos + trueNeg
    denominator = truePos + trueNeg + falsePos + falseNeg
    return numerator/denominator

def sensitivity(truePos, falseNeg):
    try:
        return truePos/(truePos + falseNeg)
    except ZeroDivisionError:
        return float('nan')

def specificity(trueNeg, falsePos):
    try:
        return trueNeg/(trueNeg + falsePos)
    except ZeroDivisionError:
        return float('nan')

def posPredVal(truePos, falsePos):
    try:
        return truePos/(truePos + falsePos)
    except ZeroDivisionError:
        return float('nan')

def negPredVal(trueNeg, falseNeg):
    try:
        return trueNeg/(trueNeg + falseNeg)
    except ZeroDivisionError:
        return float('nan')

def getStats(truePos, falsePos, trueNeg, falseNeg, toPrint = True):
    accur = accuracy(truePos, falsePos, trueNeg, falseNeg)
    sens = sensitivity(truePos, falseNeg)
    spec = specificity(trueNeg, falsePos)
    ppv = posPredVal(truePos, falsePos)
    if toPrint:
        print(' Accuracy =', round(accur, 3))
        print(' Sensitivity =', round(sens, 3))
        print(' Specificity =', round(spec, 3))
        print(' Pos. Pred. Val. =', round(ppv, 3))
    return (accur, sens, spec, ppv)

#page 409 of book Figure 24.5 Build examples and divide data into training and test sets
def divide80_20(examples):
    sampleIndices = random.sample(range(len(examples)), len(examples)//5)
    trainingSet, testSet = [], []
    for i in range(len(examples)):
        if i in sampleIndices:
            testSet.append(examples[i])
        else:
            trainingSet.append(examples[i])
    return trainingSet, testSet

#page 422 of book Figure 24.15 Use logistic regression to predict Gender
def applyModel(model, testSet, label, prob = 0.5):
    #Create vector containing feature vectors for all test examples
    testFeatureVecs = [e.getScaledFeatures() for e in testSet] #getFeatures or getScaledFeatures as necessary
    probs = model.predict_proba(testFeatureVecs)
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for i in range(len(probs)):
        if probs[i][1] > prob:
            if testSet[i].getLabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else:
            if testSet[i].getLabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg

#page 424 of book Figure 24.16 Construct ROC curve and find AUROC
def buildROC(model, testSet, label, title, plot = True):
    xVals, yVals = [],[]
    p = 0.0
    while p <= 1.0:
        truePos, falsePos, trueNeg, falseNeg = applyModel(model, testSet, label, p)
        xVals.append(1.0 - specificity(trueNeg, falsePos))
        yVals.append(sensitivity(truePos, falseNeg))
        p += 0.01
    auroc = sklearn.metrics.auc(xVals, yVals, True)
    if plot:
        pylab.plot(xVals, yVals)
        pylab.plot([0,1],[0,1], '--')
        pylab.title(title + ' (AUROC = ' + str(round(auroc, 3)) + ')')
        pylab.xlabel('1 - Specificity')
        pylab.ylabel('Sensitivity')
    return auroc
```

t

Object-oriented python code

10. SQL Queries

Link to full SQL database scripts

Table and View Creation

```
create table FipCodeToCounty (
        id int AUTO_INCREMENT primary key,
        state char(2),
        state_fip char(2),
        county_fip char(3),
        county varchar(60),
        class_fip varchar(5)
);

create table AirQualityIndexCounty (
        id int AUTO_INCREMENT primary key,
        state char(2) not null,
        fip varchar(6) not null,
        days_aqi int(3),
        good_days int(3),
        moderate_days int(3),
        sensitive_groups_days int(3),
        unhealthy_days int(3),
        very_unhealthy_days int(3),
        hazardous_days int(3),
        max_aqi int(3),
        AQI_ninety_percentile int(3),
        median_aqi int(3),
        days_co int(3),
        days_no2 int(3),
        days_ozone int(3),
        days_so2 int(3),
        days_pm25 int(3),
        days_pm10 int(3)
);

create table HospitalQualityRankings (
        id int AUTO_INCREMENT primary key,
        hospital_name varchar(100) not null,
        city varchar(60),
        state char(2) not null,
        fip varchar(6) not null,
        hospital_type varchar(40),
        hospital_ownership varchar(60),
        hospital_rating int(2)
);
```

```
create table CountyHealthRankings (
        id int AUTO_INCREMENT primary key,
        state char(2) not null,
        county varchar(60) not null,
        fip varchar(6) not null,
        premature_deaths int(12),
        fair_health_value float,
        poor_physical_health_days_value float,
        poor_mental_health_days_value float,
        low_birthweight float,
        adult_smoking_value float,
        adult_obesity_value float,
        physical_inactivity_value float,
        alcohol_impaired_driving_deaths_value float,
        unemployment_value float,
        income_inequality_value float,
        air_pollution_particulate_matter_value float,
        drug_overdose_deaths_value float,
        health_care_costs_value int(12),
        median_household_income_value int(12),
        population_estimate_value int(12),
        percent_pop_below_18 float,
        percent_pop_above_65 float
);

CREATE VIEW CombinedData AS
SELECT
AirQualityIndexCounty.median_aqi, AirQualityIndexCounty.AQI_ninety_percentile,
CountyHealthRankings.fip, CountyHealthRankings.state, CountyHealthRankings.county,
CountyHealthRankings.median_household_income_value, CountyHealthRankings.premature_deaths,
CountyHealthRankings.population_estimate_value, CountyHealthRankings.low_birthweight,
HospitalQualityRankings.hospital_rating
FROM
AirQualityIndexCounty
LEFT JOIN CountyHealthRankings ON AirQualityIndexCounty.fip = CountyHealthRankings.fip
LEFT JOIN HospitalQualityRankings ON AirQualityIndexCounty.fip = HospitalQualityRankings.fip;
```

Visualization SQL for Choropleth

```
SELECT avg(distinct(premature_deaths)/population_estimate_value)*100, state FROM CombinedData group by state

SELECT avg(distinct(low_birthweight))*100, state FROM CombinedData group by state

SELECT round(avg(DISTINCT(median_household_income_value))), state FROM CombinedData group by state

SELECT avg(hospital_rating), state FROM CombinedData group by state
```

SELECT round(avg(distinct(median_aqi))), state FROM CombinedData group by state

SELECT round(avg(distinct(AQI_ninety_percentile))), state FROM CombinedData group by state

Analysis SQL

SELECT avg(hospital_rating), fip FROM CombinedData group by fip HAVING AVG(hospital_rating) >= 0

SELECT fip, state, county, median_aqi, AQI_ninety_percentile, median_household_income_value, premature_deaths, population_estimate_value, low_birthweight FROM CombinedData where premature_deaths is not NULL and low_birthweight is not NULL

References

About Air Data Reports. (2018, September 21). Retrieved from https://www.epa.gov/outdoor-air-quality-data/about-air-data-reports

Andermann, A. (2016). Taking action on the social determinants of health in clinical practice: A framework for health professionals. *Canadian Medical Association Journal, 188*(17-18). doi:10.1503/cmaj.160177

Correlation (Pearson, Kendall, Spearman). (n.d.). Retrieved December 10, 2018, from https://www.statisticssolutions.com/correlation-pearson-kendall-spearman/

Guttag, J. (2017). Introduction to computation and programming using Python: With application to understanding data. Cambridge, MA: The MIT Press.

Hospital Compare overall hospital rating. (n.d.). Retrieved September 16, 2018, from https://www.medicare.gov/hospitalcompare/About/Hospital-overall-ratings.html

Kelly, F. J., & Fussell, J. C. (2015). Air pollution and public health: Emerging hazards and improved understanding of risk. *Environmental Geochemistry and Health, 37*(4), 631-649. doi:10.1007/s10653-015-9720-1

Larrimore, J. (2011). Does a Higher Income Have Positive Health Effects? Using the Earned Income Tax Credit to Explore the Income-Health Gradient. *Milbank Quarterly,89*(4), 694-727. doi:10.1111/j.1468-0009.2011.00647.x

Parrish, R. G., MD. (2010). Preventing Chronic Disease: July 2010: 10_0005. Retrieved September 18, 2018, from https://www.cdc.gov/pcd/issues/2010/jul/10_0005.htm

Swift, E. K. (2002). *Guidance for the national healthcare disparities report*. Retrieved September 17, 2018, from https://www.ncbi.nlm.nih.gov/books/NBK221045/

Data Sources Guide

- **FIP Code Reference Key**
    - US Zipcode to County State to FIPS Look Up - dataset by niccolley. (2018, May 28). Retrieved October 13, 2018, from https://data.world/niccolley/us-zipcode-to-county-state
- **Health rankings/outcomes by region**
    - Explore Health Rankings. (n.d.). Retrieved September 10, 2018, from http://www.countyhealthrankings.org/explore-health-rankings
- **Hospital quality ratings**
    - Hospital General Information (2017). Retrieved September 13, 2018 from https://www.kaggle.com/cms/hospital-general-information/home
- **State Reference Key**
    - Admin. (2015, November 09). List of 50 US States in Excel. Retrieved October 02, 2018, from https://scottontechnology.com/list-of-50-us-states-in-excel/
- **US Air pollution levels**
    - US Facility-Level Air Pollution (2010-2014). Retrieved September 13, 2018 from https://www.kaggle.com/jaseibert/us-facilitylevel-air-pollution-20102014