SI 618 Project 1 Report

Comparing Health Searches to Disease Prevalence by U.S. Cities

Kim Vuong
kvuong

I. MOTIVATION

With an interest in public health, I wanted to look at data in U.S. cities relating to population health and information. For this project, I looked at one dataset that contains participant responses to a CDC survey on health outcomes, prevention, and unhealthy behavior. The other dataset contains Google health searches by U.S. metropolitan areas from 2005-2017.

I was interested in seeing if there was a trend between areas with a higher increase in specific health searches and areas with a high related disease prevalence. Hence, I focused on disease prevalence from the CDC data and picked a time span of three years from the Google health searches dataset for my analysis.

II. DATA SOURCES

A. **"Health searches by US Metropolitan Area, 2005-2017" -** https://www.kaggle.com/GoogleNewsLab/health-searches-us-county

   This dataset is available on Kaggle in a CSV format with the original source coming from Google. The file contains a total of 210 records with 128 columns for the years spanning 2005-2017 and 9 Google health search topics for each year. These health topics include cancer, cardiovascular, stroke, depression, rehab, vaccine, diarrhea, obesity, and diabetes. For my project, I selected cancer, cardiovascular, and diabetes as the search topics to focus on. For the year, I choose a 3-year time span counting back from what the CDC disease prevalence dataset had reported for the corresponding disease or health condition for the health search. For cardiovascular searches, I used the data columns for the years 2012-2015, with high blood pressure prevalence data being reported for 2015. For cancer and diabetes searches, I used the data columns for the years 2013-2016 since the CDC data reported prevalence of these diseases for 2016.

B. **"CDC 500 Cities"** - https://www.kaggle.com/cdc/500-cities

   This dataset is also available on Kaggle in a CSV format with the original source coming from the Centers of Disease Control and Prevention. The data is reported as percentages of residents responding affirmative ("Yes") to the health-related questions by city. There are 500 records with 119 columns. The 119 columns include crude prevalence, age-adjusted prevalence, confidence intervals for both measures for each surveyed question under the main three topics of health outcomes, prevention, and unhealthy behavior. For my project, the chose to focus on "high blood pressure among adults aged>=18 years -2015", "cancer (excluding skin cancer) among adults aged >=18 years -2016", and "diagnosed diabetes among adults aged >=18 years – 2016". I selected the age-adjusted prevalence for each of the mentioned categories so that the differences in age distribution depending on the area will not be a factor in influencing the prevalence rate (e.g. Fort Myers, FL has a much higher population of elderly people since it is a place for retirement).

III. DATA MANIPULATION METHODS

A. Cleaning the Dataset

The “CDC 500 Cities” dataset did not require any cleaning, so I used that dataset as is. For the “Health searches by US Metropolitan Area, 2005-2017” dataset, some of the places listed were just the city and state abbreviation, while others were a combination of cities between 2-4 cities to represent an area (e.g. Greensboro-High Point-Winston Salem NC). Using pandas, I decided to remove all records with a combination of cities, which is noted by a ‘-‘ in the string (code used: no_multicities_df = df[~df.dma.str.contains('-')]). Further explanation on why I decided to do this is under **Section V. Challenges**. Then, to match the format of the “CDC 500 Cities” dataset so that I can more easily do SQL queries later, I split the city and state string on the last space of the string by indexing (code used: city_state_split_df= no_multicities_df['dma'].str.rsplit(' ', 1,expand=True)). This is because some city names are two or even three words long. Since the data is formatted with the city name, a space, and then a state abbreviation instead of having the usual comma to separate the city name from the state, I had to split it using this method to ensure I get one column as the full city name and the other as just the state abbreviation. I removed the original column [‘dma’] and replaced it with the two columns I created to the dataframe. Since, I did not have any missing, incomplete, or incorrect data, I wrote the cleaned dataset to a new CSV file.

B. SparkSQL to Run Queries and Create Dataframes and Tables

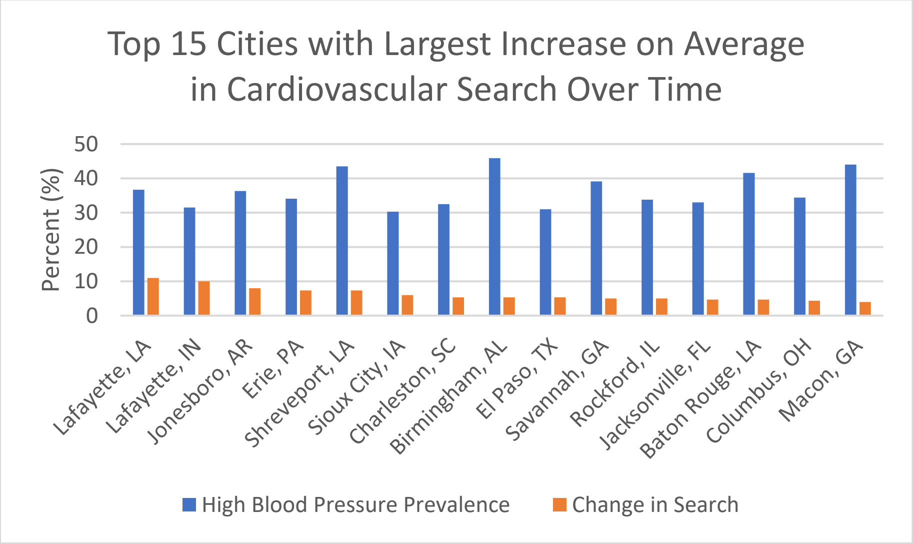
I created a dataframe for each of the relevant data for my project from each of the two datasets. Using the sqlContext.read.csv() command, I read in both datasets as dataframes and registered each to a table using registerTempTable(). From there, I created another two tables by selecting from each of the two tables created from the loaded data from the CSV files. Using sqlContext.sql(), I selected the state abbreviation, city name, and prevalence for high blood pressure, cancer, and diabetes and created one table. For the other table, I selected the state abbreviation, city name, and all the years+health search topic that was relevant to my project. From here, I 1) queried for the state abbreviation, city, and prevalence for high blood pressure and created a table for it, 2) queried for the state abbreviation, city, the four relevant health search column years and created a table for it and 3) joined the two tables on the state abbreviation and city and saved it to a variable to be used for the computational tasks. I repeated these three steps for the cancer and diabetes data.

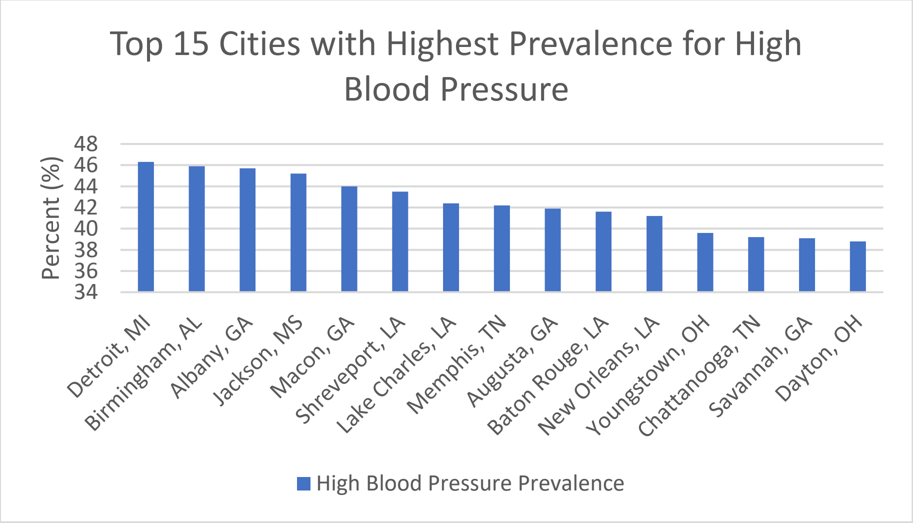
IV. ANALYSIS AND VISUALIZATION

A. Computational Task 1: What are the top 15 cities with the largest increase in cardiovascular searches on average?

To answer this, I needed to calculate the difference between each year's search percentage and then find the average of all three differences. I converted my SQL query with the joined tables to an RDD by using the .rdd command. To calculate the difference between the years, I used the .map() command with the lambda function to do the subtraction. Because the data are in strings in tuples format, I had to convert the numbers to a float type to carry out the math (code used: rdd1 = q5.rdd.map(lambda x: (x[0], x[1], x[2], float(x[4]) - float(x[3]), float(x[5]) - float(x[4]), float(x[6]) - float(x[5])))). Next, to find the average of the differences, I used .map() again with the lambda function to add up the three values calculated from the previous step and divide it by 3 (code used: avg_rdd1 = rdd1.map(lambda x: (x[0], x[1], x[2], (x[3] + x[4] + x[5])/3))). I then sorted the data by the highest average increase by using the .sortBy() command with the lambda function (code used: sorted_rdd1 = avg_rdd1.sortBy(lambda x: x[3], ascending=False)). After that, I formatted the data to be tab-separated by using the .map() command with the lambda function inserting tabs between each element and converting the average increase element back to a string (code used: formatted_rdd1 = sorted_rdd1.map(lambda x: x[0] + '\t' + x[1] + '\t' + x[2] + '\t' + str(x[3]))). After merging the output file to a .tsv file, the format of the .tsv file looks as followed: state abbreviation, city, high blood pressure prevalence, average increase in percent in cardiovascular search.

I graphed the top 15 cities with the largest increase in cardiovascular searches along with its high blood pressure prevalence using Excel and to compare, the top 15 with highest prevalence.



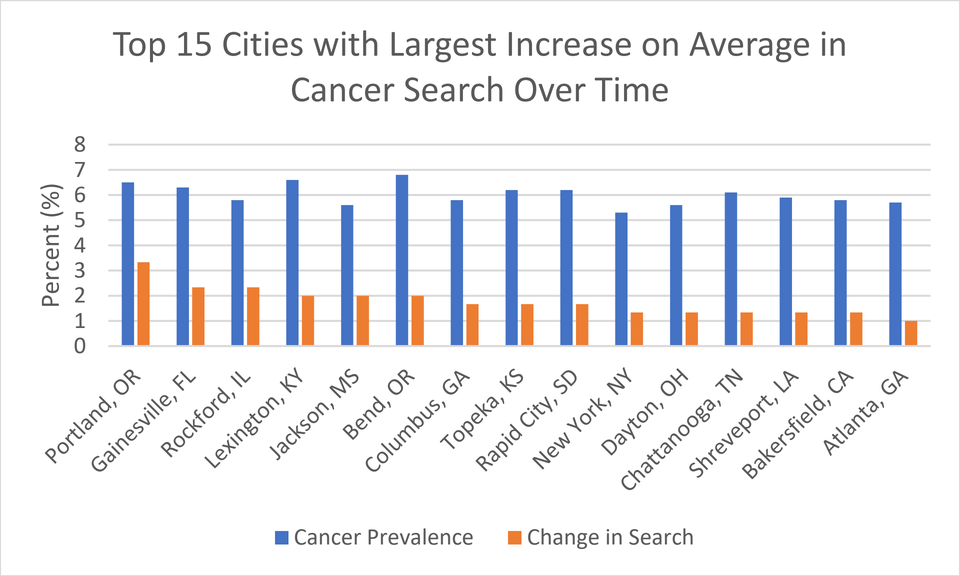


A total of 5 cities were on both graphs (Birmingham, AL; Savannah, GA,; Baton Rouge, LA; Macon, GA; Shreveport, LA). These all happen to be cities located in the Southern region.

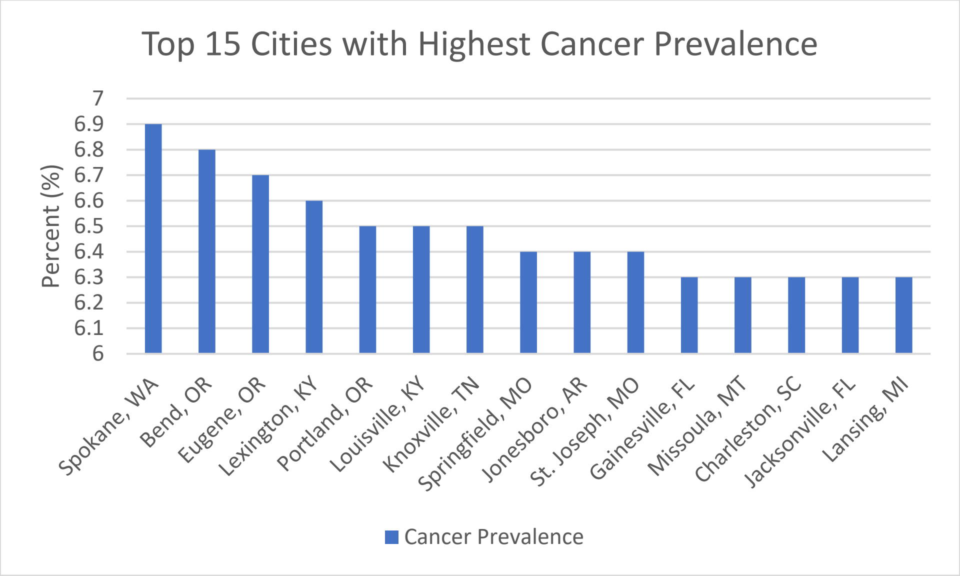
B. Computational Task 2: What are the top 15 cities with the largest increase in cancer searches on average?

To answer this, I needed to calculate the difference between each year's search percentage and then find the average of all three differences. I converted my SQL query with the joined tables to an RDD by using the .rdd command. To calculate the difference between the years, I used the .map() command with the lambda function to do the subtraction. Because the data are in strings in tuples format, I had to convert the numbers to a float type to carry out the math (code used: rdd2 = q8.rdd.map(lambda x: (x[0], x[1], x[2], float(x[4]) - float(x[3]), float(x[5]) - float(x[4]), float(x[6]) - float(x[5])))). Next, to find the average of the differences, I used .map() again with the lambda function to add up the three values calculated from the previous step and divide it by 3 (code used: avg_rdd2 = rdd2.map(lambda x: (x[0], x[1], x[2], (x[3] + x[4] + x[5])/3))). Next, I sorted the data by the highest average increase by using the .sortBy() command with the lambda function (code used: sorted_rdd2 = avg_rdd2.sortBy(lambda x: x[3], ascending=False)). After that, I formatted the data to be tab-separated by using the .map() command with the lambda function inserting tabs between each element and converting the average increase element back to a string (code used: formatted_rdd2 = sorted_rdd2.map(lambda x: x[0] + '\t' + x[1] + '\t' + x[2] + '\t' + str(x[3]))). After merging the output file to a .tsv file, the format of the .tsv file looks as followed: state abbreviation, city, high blood pressure prevalence, average increase in percent in cardiovascular search.

I graphed the top 15 cities with the largest increase in cancer searches along with its cancer prevalence using Excel and to compare, the top 15 with highest prevalence.



Top 15 Cities with Largest Increase on Average in Cancer Search Over Time



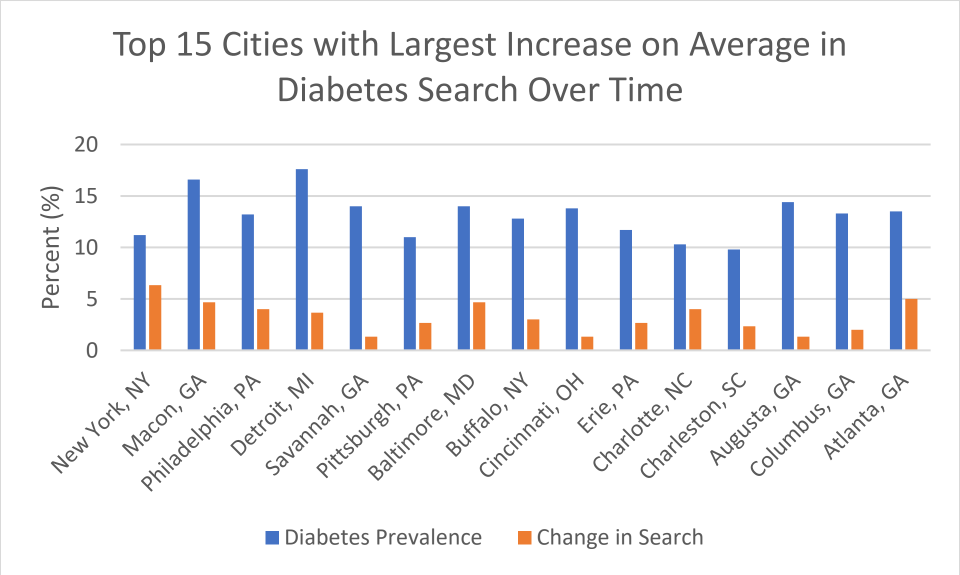
Top 15 Cities with Highest Cancer Prevalence

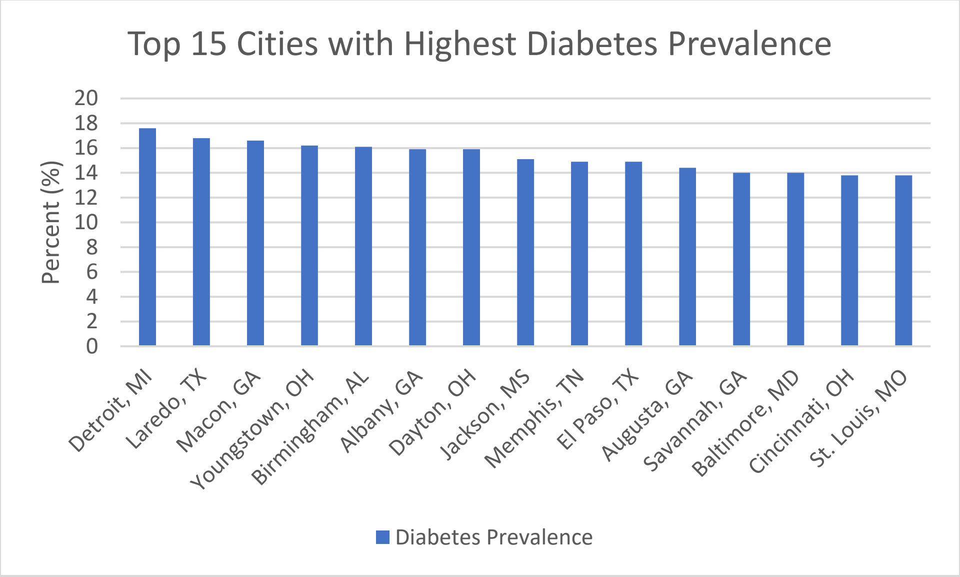
A total of 4 cities were on both graphs (Portland, OR; Gainesville, FL; Lexington, KY; Bend, OR). These cities were either in the Southern region or located in OR. Interestingly, 4 out of the top 5 cities with the highest cancer prevalence are in the Pacific Northwest region.

C. Computational Task 3: What are the top 15 cities with the largest increase in diabetes searches on average?

To answer this, I needed to calculate the difference between each year's search percentage and then find the average of all three differences. I converted my SQL query with the joined tables to an RDD by using the .rdd command. To calculate the difference between the years, I used the .map() command with the lambda function to do the subtraction. Because the data are in strings in tuples format, I had to convert the numbers to a float type to carry out the math (code used: rdd3 = q11.rdd.map(lambda x: (x[0], x[1], x[2], float(x[4]) - float(x[3]), float(x[5]) - float(x[4]), float(x[6]) - float(x[5])))). Next, to find the average of the differences, I used .map() again with the lambda function to add up the three values calculated from the previous step and divide it by 3 (code used: avg_rdd3 = rdd3.map(lambda x: (x[0], x[1], x[2], (x[3] + x[4] + x[5])/3))). Next, I sorted the data by the highest average increase by using the .sortBy() command with the lambda function (code used: sorted_rdd3 = avg_rdd3.sortBy(lambda x: x[3], ascending=False)). After that, I formatted the data to be tab-separated by using the .map() command with the lambda function inserting tabs between each element and converting the average increase element back to a string (code used: formatted_rdd3 = sorted_rdd3.map(lambda x: x[0] + '\t' + x[1] + '\t' + x[2] + '\t' + str(x[3]))). After merging the output file to a .tsv file, the format of the .tsv file looks as followed: state abbreviation, city, high blood pressure prevalence, average increase in percent in cardiovascular search.

I graphed the top 15 cities with the largest increase in diabetes searches along with its diabetes prevalence using Excel and to compare, the top 15 with highest prevalence.





A total of 5 cities were on both graphs (Detroit, MI; Macon, GA; Augusta, GA; Savannah, GA; Cincinnati, OH). Georgia seems to have a high rate of diabetes with 4 cities being in the top 15.

From the results of these 3 tasks, about 1/3 of the top 15 cities with the highest increase in a related health search are also in the top 15 most disease prevalent. From the visualizations, I noticed there seems to be certain same cities that have a high prevalence for high blood pressure, cancer, and diabetes.

IV. CHALLENGES

A. Cleaning Dataset

For the health search dataset, initially I was going to try to split the combination cities into new rows. However, the format was in a string and not a list, so I would not be able to use pandas.DataFrame.explode(). The combined city names ranged between 2-4 different cities and I could not think of a way to split on the hyphens '-' and still be able to get the state abbreviation to stay intact with each of the cities. Also, a concern was even if I was able to split the combination into new cities, there will be duplicate data since the original data was aggregate for all those cities. This could affect the end results of the analysis.

Resolution: I removed all rows containing combination cities from the dataset.

B. Schema not showing values when selecting the 'year+search_topic' columns

The column names for the search topics are in the format year+search_topic. I kept getting error messages when I tried to select those columns. I tried putting double quotes around them, but then, they would not show the values in the schema. Instead, where the values were supposed to be, it was the column name. I knew it was not single quotes because the SQL query statements as whole has to be in single quotes, so subparts within the statements cannot be single quotes.

Resolution: I consulted Scott during office hours and he suggested use the backtick key, which worked.

C. Error when creating output file

I kept getting an error when I tried to run .saveAsTextFile(). All of the .map() lambda functions prior ran without any errors. When I looked at the error message it said TypeError: unsupported operand type(s) for -: ‘unicode’ and ‘unicode’.

Resolution: I went back and tried to debug my .map() lambda functions where the calculations were being executed. I used the .take() command to see if the calculations are being performed correctly after each step. I found that the syntax I used for converting the string element to float was not correct and I tried to use the sum() method for 3 elements but a quick google search said it can only take up to two parameters. I adjusted by using () and the addition sign so that this calculation is done first and then divided by 3, the total number of values, to get the average difference.