**Jacob Thomas ETL Project for UT Data Science Bootcamp**

**Proposal:**

A colleague of mine is interested in investigating the hypothesis that proximity of geospatial location of residence to a high-density of tobacco outlets is associated with a higher likelihood of having smoked cigarettes in the previous 30 days, and that geospatial location of residence to a high-density of alcohol outlets is associated with a higher likelihood of having consumed alcohol in the previous 30 days.

She is a marketing scientist, and luckily, she already has a national repository of data that lists all registered tobacco and alcohol retailers, including bars, liquor stores, grocery stores / supermarkets, smoke / vape shops, drug stores / pharmacies, and convenience stores. But, she does not have individual-level outcomes related to smoking and drinking and where the individual resides. She was thrilled to learn that my lab has the exact outcome data that she needs. My existing data surveys a sample of over 5000 young adults on mental health and substance use data over time, with a total of 7 timepoints collected. Beginning at timepoint #7, we collected location data on the participants which enables this collaboration to happen.

However, when collaborating on initial data exploration, we encountered a problem! The geospatial analysis software that would be used in this study requires latitude and longitude coordinates, and the only location variable that my dataset has is zip codes.

Initially, she dreaded having to comb through thousands of individual’s zip codes and manually look up an associated latitude longitude coordinate. Fortunately, I have recently learned (**thanks to the UT Austin Bootcamp)** how to use publicly available API’s to make location calls and transform zip codes into coordinates.

The present data science project will use Python to ***extract*** data from two sources: a large epidemiological data set of substance use and mental health outcomes of individuals, and a dataset generated in real time by making calls to a publicly available API that can extrapolate coordinates from zip code (OpenWeatherAPI). Once all necessary data is extracted, the two datasets will be ***transformed*** into one succinct file with only the variables needed to perform the analysis that will answer my colleague’s hypotheses. Finally, the file will be ***loaded*** as two files, a completed CSV and an SQL.

**Technical Report:**

>> Please review the accompanying Jupyter notebook as a complement to the technical report.

* The ***extracted*** source files are as follows:

1. A proprietary research dataset consisting of >5000 emerging adults in Texas at the time of recruitment (metro areas of Austin, DFW, Houston, and San Antonio) which was collected at The University of Texas and funded by the Centers for Disease Control. The data surveys demographics, substance use, educational outcomes, health status, and subjective wellbeing, which were collected in a time-series format, every 6-month, currently at a total of 7 timepoints (waves). The most recent wave collected each participant’s current location of residence. Since this is a longitudinal dataset, it is fair to assume that residence may have changed since the initial wave, so this data is only validly associated with other data from the most recent wave.
2. A second dataset was generated in real time using calls from the OpenWeatherAPI. The OpenWeatherAPI is a flexible tool that is intended to search for live weather reports based on location; however with some digging, I discovered a unique feature that is useful for my purposes. When making a call for a live weather report based on zip code, a JSON file is returned that includes coordinates at the center of that zip code!

For example, making an API call for zip code 78757 returns:

{"coord":{"lon":-97.66,"lat":30.37},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":86.86,"pressure":1016,"humidity":66,"temp\_min":84,"temp\_max":90},"visibility":16093,"wind":{"speed":11.41,"deg":210,"gust":7.2},"clouds":{"all":1},"dt":1565624190,"sys":{"type":1,"id":5739,"message":0.0124,"country":"US","sunrise":1565610940,"sunset":1565658954},"timezone":-18000,"id":4737316,"name":"Austin","cod":200}

I created a 2nd dataset with zip code from my original data, and with latitude and longitude columns extracted using the API’s JSON files.

* Once my data sources were read in my Jupyter notebook, they were ***transformed***.
  + First, I used a merge function to join the two datasets based on the common variable (zip code).

>> df3 = pd.merge(df, df2, on='zip\_8')

* + Then, I inspected the dataset and dropped all columns that are irrelevant to the proposed study.

>> df4 = df3.drop(columns= *[…, …, …]*)

* + Finally, I dropped all rows with missing data.

>> df5 = df4[df4.Latitude != 'NaN']

* The last step of the ETL process was to ***load*** the final dataset. I generated new CSV and SQL files using the following commands:

>> df5.to\_csv('Post-ETL-Data.csv', index=False)

>> from sqlalchemy import create\_engine

engine = create\_engine('sqlite://', echo=False)

df5.to\_sql('Post-ETL-Data-SQL', con=engine)

**Final Report:**

* **E**xtract: 2 data sources from an existing CSV, and JSONs called using an API.
* **T**ransform: I imported stored array data into columns of a dataframe, merged two dataframes onto a common variable, inspected for continuity, and cleaned to remove missing and irrelevant data.
* **L**oad: The final data was loaded in CSV and SQL. The CSV was generated so that statistical software and the geospatial analysis software we will use can easily import the data, and the SQL was generated to allow for on-the-fly querying if we wanted to look at specific things without a statistical analysis.