**Air Quality Comparison**

**San Diego vs. Los Angeles vs. Denver vs. Atlanta vs. Nashville**

**From the coast, to the mountains, to the forest…**

**Methodology**

**Data Collection**

1. Collection of pollen data was achieved using webscraping of [www.pollen.com/research/](http://www.pollen.com/research/). The data scraped encompassed tree, grass and ragweed plants documented to grow in each city. The pollens associated with these are categorized using flowering seasons (Spring, Summer, Fall and Winter), pollen type (Tree, Grass, Ragweed) and allergenicity levels (Mild, Moderate or Severe.) This data was saved as a DataFrame that was exported as a CSV file.

Challenges: A new module, Selenium, was necessary to iterate through the different chart categories (i.e. Season and Pollen Category). This module was installed using the Anaconda platform and documentation of Selenium researched to simulate ‘mouse-click’ behavior within the website.

Tech used: Webscraping, Python, Anaconda platform, Modules: pandas, requests, BeautifulSoup, Selenium

1. Collection of general city information was achieved using webscraping of <https://en.wikipedia.org/wiki/>. The categories gleaned included city name, county, elevation, population census year, population density, metro population, population rank and climate type. This data was saved as a DataFrame that was exported as a CSV file.

Challenges:

* The wiki data was not consistently tagged or titled (i.e. several states had uniquely titled html tags and some population fields were mislabeled.)
* Climate information was embedded in links in the body of the page text.
* Some cities had multiple climates listed, which were consolidated using the most common wording to optimize comparison.
* Some counties had unnecessary symbols or words.
* UNICODE characters were embedded in the elevation data.
* The population and elevation data included both metric and standard measurements, where only standard measurements were necessary for this project.
* Some elevations contained a range of values instead of a single value.

All issues were handled in the webscraping code using conditional statements, string manipulation (splitting, slicing, concatenation and character replacement), and averages of the range of values. The resulting DataFrame retained only one potential issue of a city having multiple associated counties, which \*\*could\*\* impact its ability to merge efficiently with other tables.

Tech used: Webscaping, Python, Python modules: requests, pandas, numpy, bs4 (BeautifulSoup), and regex

1. Collection of particulate pollution data began with understanding the AQS Data Dictionary, available as a PDF at <https://www.epa.gov/aqs/aqs-data-dictionary>.

I then looked at understanding the measurement parameters for the EPA AQS using the AQS Reference Table, available at <https://aqs.epa.gov/aqsweb/documents/codetables/parameters.html>. The 1,477 available parameters were narrowed down to \_\_\_13\_\_ by filtering for measurements based on the following criteria:

* 1. Compounds including ground level ozone (O3), particle pollution (PM10 and PM2.5), carbon monoxide (CO), nitrogen dioxide (NO2), or sulfur dioxide (SO2) which are “the most common ambient air pollutants regulated under the Clean Air Act” (source: <https://www.epa.gov/pmcourse/patient-exposure-and-air-quality-index>).
  2. Compounds including smoke and carbon dioxide (CO2).
  3. Compounds that were valid according to the ‘Still Valid’ field.
  4. Overall PM2.5 and PM10 measurements

In order to follow EPA’s guidelines, time between requests was kept to a minimum of 5 seconds. This prevented any disabling of my account due to violation of their Terms of Service.