COVID 19 Effects on Air Quality

Daniel Love, Nancy Mathur, Angie Ong & Chad Gielow

2020

DATA MUNGING

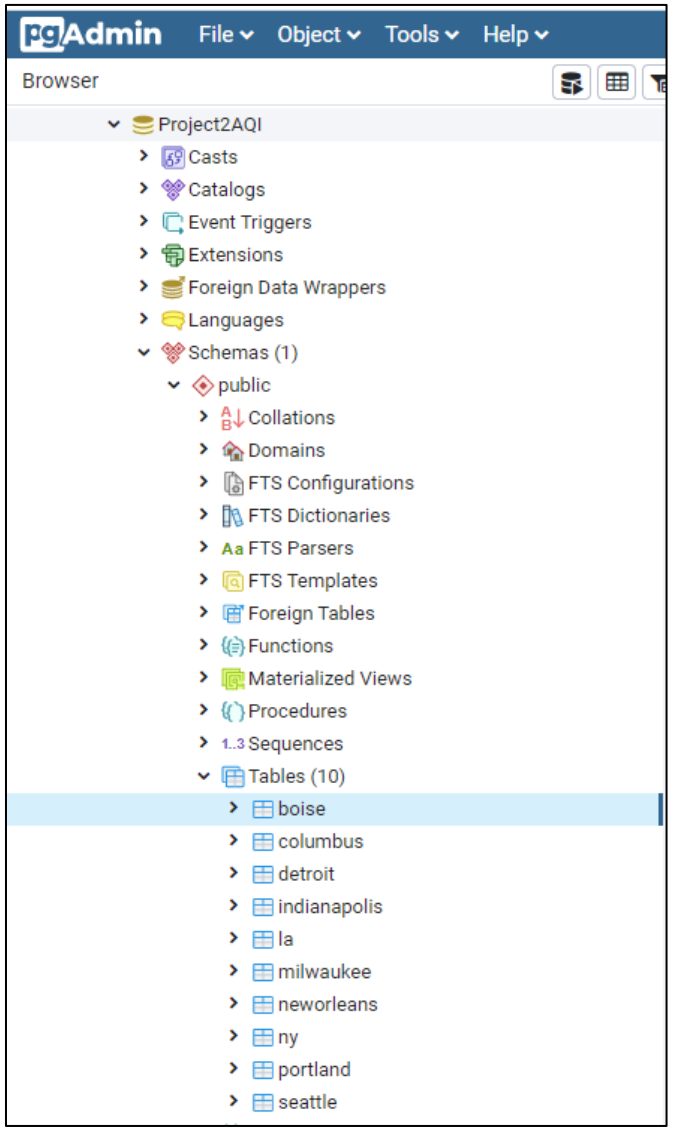
Cleaning all 10 csv files for each city

In [4]:
```
# Add latitude and longitude for LA
dataLA["lat"] = 34.0522
dataLA["lon"] = -118.2437

# Add City name
dataLA["city_name"] = "Los_Angeles"

# Add shelter in place date
dataLA["state_ordinance"] = "2020-03-23"

#Add population
dataLA["population"] = int("3990456")

# Changed date colume from object to datetime
dataLA["Date"] = pd.to_datetime(dataLA["Date"])
dataLA["state_ordinance"] = pd.to_datetime(dataLA["state_ordinance"])

# Rename columns
dataLA = dataLA.rename(columns= {"Date": "date",
                                 "Overall_AQI_Value": "overall_aqi_value",
                                 "Main Pollutant": "main_pollutant",
                                 "Site Name (of Overall AQI)": "site_name",
                                 "Site ID (of Overall AQI)": "site_id",
                                 "Source (of Overall AQI)": "source_aqi",
                                 "CO": "co",
                                 "Ozone": "ozone",
                                 "SO2": "so2",
                                 "PM10": "pm10",
                                 "PM25": "pm25",
                                 "NO2": "no2"})
```

- Jupyter Notebook to clean and Mung CSV files
- Read files
- Update column names
- Join Data

DATABASE

```
In [17]: # To load to postgreSQL and add date as PK to each table
dataLA.to_sql("la", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE la ADD PRIMARY KEY (date);")

dataBoise.to_sql("boise", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE boise ADD PRIMARY KEY (date);")

dataColumbus.to_sql("columbus", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE columbus ADD PRIMARY KEY (date);")

dataDetroit.to_sql("detroit", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE detroit ADD PRIMARY KEY (date);")

dataMilwaukee.to_sql("milwaukee", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE milwaukee ADD PRIMARY KEY (date);")

dataNewOrleans.to_sql("neworleans", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE neworleans ADD PRIMARY KEY (date);")

dataNY.to_sql("ny", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE ny ADD PRIMARY KEY (date);")

dataPortland.to_sql("portland", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE portland ADD PRIMARY KEY (date);")

dataSeattle.to_sql("seattle", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE seattle ADD PRIMARY KEY (date);")

dataIndianapolis.to_sql("indianapolis", con=conn, if_exists="replace", index=False)
with engine.connect() as con:
    con.execute("ALTER TABLE indianapolis ADD PRIMARY KEY (date);")
```



- Jupyter Notebook to create Primary Keys
- Load data into postgresDB
- Use postgresDB to query information for the Flask App API

FLASK APP- API

```
# Database Setup
#################################################

# engine = create_engine("postgresql://postgres:postgres@localhost:5432/Project2AQI")
engine = create_engine(f"postgresql://postgres:{password}@localhost:5432/Project2AQI")

conn = engine.connect()

Base = automap_base()
# reflect the tables
Base.prepare(engine, reflect=True)

# Save reference to the table
Boise = Base.classes.boise
Columbus = Base.classes.columbus
Detroit = Base.classes.detroit
Milwaukee = Base.classes.milwaukee
La = Base.classes.la
Neworleans = Base.classes.neworleans
Ny = Base.classes.ny
Portland = Base.classes.portland
Seattle = Base.classes.seattle
Indianapolis = Base.classes.indianapolis

```

```
    # Create a dictionary to hold boise data
    boise_aqi = []
    for date, overall_aqi_value, main_pollutant, site_name, site_id,
        boise_dict = {}
        boise_dict["date"] = date
        boise_dict["aqi_value"] = overall_aqi_value
        boise_dict["main_pollutant"] = main_pollutant
        boise_dict["site_name"] = site_name
        boise_dict["site_id"] = site_id
        boise_dict["source_aqi"] = source_aqi
        boise_dict["lat"] = lat
        boise_dict["lon"] = lon
        boise_dict["city_name"] = city_name
        boise_dict["state_ordinance"] = state_ordinance
        boise_dict["population"] = population
        boise_aqi.append(boise_dict)

    return jsonify(boise_aqi)

@app.route("/api/v1.0/columbus")
def columbus():
    # Create our session (link) from Python to the DB
    session = Session(engine)
```

- Python to code the FLASK APP
- Link with postgresDB
- Reference each table
- Return data to index.HTML
- Create dictionary for each city to hold data

HTML

```
<body>
  <!-- Overall conatainer -->
  <div class="container-fluid">
    <!-- ROW 1 Navbar-->
    <div class="row stick-top">
      <!-- Navbar -->
      <nav class="navbar navbar-expand-lg navbar-light theheader">
        <!-- Page Title -->
        <a class="navbar-brand text-white" href="http://127.0.0.1:5500/templates/index.html">Covid 19 Effe
        <!-- Hamburger toggler button -->
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedC
          <span class="navbar-toggler-icon hamburger"></span>
        </button>
        <!-- City dropdown links to visuals -->
        <div class="collapse navbar-collapse navvy_bg" id="navbarSupportedContent">
          <ul class="navbar-nav col-lg-4 col-auto">
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle
                  City Data
                  </a>
              <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                  <a class="dropdown-item" href="http://127.0.0.1:5000/boise">Boise</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/columbus">Columbus</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/detroit">Detroit</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/indianapolis">Indianapolis</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/milwaukee">Milwaukee</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/la">Los Angeles</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/neworleans">New Orleans</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/ny">New York City</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/portland">Portland</a>
                  <a class="dropdown-item" href="http://127.0.0.1:5000/seattle">Seattle</a>
              </div>
            </li>
          </ul>
        </div>
```

```
    </div><!-- closes ROW 1 -->
    <!-- ROW 2 Statistical Data -->
    <div class="row">
      <div class="col-md-12">
        <div class="row">
          <div class="col-md-3 col-sm-6 data" id="dataL">
            <p>Last Years AQI Average</p>
            <hr>
            <h2 id="pre"></h2>
          </div>
          <div class="col-md-3 col-sm-6 data" id="dataML">
            <p>Post Ordinance AQI Average</p>
            <hr>
            <h2 id="post"></h2>
          </div>
          <div class="col-md-3 col-sm-6 data" id="dataMR">
            <p>Shelter in Place Ordinance</p>
            <hr>
            <h2 id="shelterDate"></h2>
          </div>
          <div class="col-md-3 col-sm-6 data" id="dataR">
            <p>City Population</p>
            <hr>
            <h2 id="population"></h2>
          </div>
        </div>
      </div>
    </div><!-- closes ROW 2 -->
    <!-- ROW 3 Graphs -->
    <div class="row row-background align-items-start">
      <!-- Bar Graph Section -->
      <div class="col-lg-6 col-md-12 bar_graph">

          <div id="chart" style="max-width: 900px; height: 300px; margin: 25px; margin-left: 0px;"></div>

      </div><!-- Closes bar graph section -->
      <!-- Line graph Secion -->
      <div class="col-lg-6 col-md-12 line_graph">
        <!-- <h3>
            Visualizations
```

- 2 HTML pages
- Index is the landing page which holds our choropleth visualization
- Used CSS to create the look and feel of our site
- Bootstrap to help build the initial framework

JAVA SCRIPT- INDEX

static > js > JS logic.js > ...

```
// Creating map object
var map = L.map("map", {
  center: [37.8, -95.5795],
  zoom: 4
});

// Adding tile layer
L.tileLayer("https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token={accessToken
  attribution: "Map data &copy; <a href=\"https://www.openstreetmap.org/\">OpenStreetMap</a
  maxZoom: 18,
  id: "mapbox.light",
  accessToken: API_KEY
}).addTo(map);


// Load in geojson data
var statesAQI = "../static/data/statesAQI.json";

// Format Date
function formatDate(nowDate) {
  nowDate = (nowDate.getMonth() + 1) +'/'+ nowDate.getDate() +"/"+ nowDate.getFullYear();
  if (nowDate === "12/31/1969")
    nowDate = "None"

  return nowDate;
}

function numberWithCommas(x) {
  return x.toString().replace(/\B(?<!\.\d*)(?=(\d{3})+(?!\d))/g, ",");
}

var geojson;
var aqi;
```

```
  // Binding a pop-up to each layer
  onEachFeature: function(feature, layer) {
    layer.bindPopup(feature.properties.City+", "+feature.properties.State +
    "<br>Population: " + numberWithCommas(feature.properties.Population) +
    "<br>Shelter in Place Ordinance: " + formatDate(new Date(feature.properties.S
    "<br>Post Ordinance Average AQI: " + feature.properties.AQI);
  }
}).addTo(map);

// Set up the legend
var legend = L.control({ position: "bottomright" });
legend.onAdd = function() {
  var div = L.DomUtil.create("div", "info legend");
  var limits = geojson.options.limits;
  var colors = geojson.options.colors;
  var labels = [];

  // Add min & max
  var legendInfo = "<h1>AQI Values</h1>" +
    "<div class=\"labels\">" +
      "<div class=\"min\">" + limits[0] + "</div>" +
      "<div class=\"max\">" + limits[limits.length - 1] + "</div>" +
    "</div>";

  div.innerHTML = legendInfo;

  limits.forEach(function(limit, index) {
    labels.push("<li style=\"background-color: " + colors[index] + "\"></li>");
  });

  div.innerHTML += "<ul>" + labels.join("") + "</ul>";
  return div;
};

// Adding legend to the map
legend.addTo(map);

});
```

- Used Leaflet library to create choropleth visual
- D3 to read the json data
- Created pop-up layer to highlight key facts about each states AQI

JAVA SCRIPT - DASHBOARD

```
    d3.json(url).then(function (data) {
        // console.log(data)
        var chosenCityDate = [];
        var chosenCityAqi = [];
        var chosenCityName = [];
        var chosenCityShelterDate = []
        var chosenCityPopulation = []

        for (i=0; i<data.length; i++) {
            chosenCityDate.push(data[i].date);
            chosenCityAqi.push(data[i].aqi_value);
            chosenCityName.push(data[i].city_name);
            chosenCityShelterDate.push(data[i].state_ordinance)
            chosenCityPopulation.push(data[i].population)

        var chosenCityName2 = chosenCityName[0]
        var chosenCityPopulation2 = chosenCityPopulation[0]
        var shelterDay = new Date(chosenCityShelterDate[0])
        var chosenCityShelterDate2 = new Date(chosenCityShelterDate[0]).toISOString(
        console.log("Angie",chosenCityName2)

        // Sort and format date
        var newDate = chosenCityDate.sort(function(a, b) {
            return  +new Date(a.date) - +new Date(b.date);
        })
        var newDate2 = [];
        for (i=0; i<newDate.length; i++) {
            newDate2.push(new Date(newDate[i]).toISOString().slice(5,10));
        }

        //  PRE and POST shelter-in-place data for the city
        var postShelterAqi = data.filter(elementData => new Date(elementData.date) >
        var preShelterAqi = data.filter(elementData => new Date(elementData.date) <=

        // PRE and POST shelter-in=place date
        var meanAqiPost = mean(postShelterAqi.map(aqidata => aqidata.aqi_value));
        var meanAqiPre = mean(preShelterAqi.map(aqidata => aqidata.aqi_value));
        var round_meanAqiPost = Math.round(meanAqiPost,2);
        var round_meanAqiPre = Math.round(meanAqiPre,2);

        // Select and input statiscal data
        document.getElementById("pre").innerHTML = round_meanAqiPre;
        document.getElementById("post").innerHTML = round_meanAqiPost;
        document.getElementById("shelterDate").innerHTML = chosenCityShelterDate2;
        document.getElementById("population").innerHTML = chosenCityPopulation2;
    }
```

```
        var updateChart = function (count) {
            count =  count || 1; // count is number of times loop runs to ge

                for (var j = 0; j < count; j++) {
                    xVal = new Date(dateThisYear[val])
                    yVal = aqiLastYear[val]
                    dps.push({
                        x: xVal,
                        y: yVal
                    });
                    val++;
                }
                for (var j = 0; j < count; j++) {
                    xVal2 = new Date(dateThisYear[val2])
                    yVal2 = aqiThisYear[val2]
                    dps2.push({
                        x: xVal2,
                        y: yVal2
                    });
                    val2++;
                }
                // if (dps.length > dataLength && dps2.length > dataLength)
                //     dps.shift();
                //     dps2.shift();
                // }
                // else if (dps.length === dataLength && dps2.length === dat
                //     dps.shift();
                //     dps2.shift();
                // }
                chart.render()
            };

            updateChart(dataLength);
            setInterval(function(){ updateChart() }, updateInterval);
            // getPlot();
    // })

}

function updateGraphs() {
    getSummary();
    getPlot();
    // getSpline();
}

updateGraphs();
```

- Visualization updated based on user city selection
- Used Echarts & CanvasJS libraries to create the visuals
- Coded math to create on page metrics
- console.log (everything)