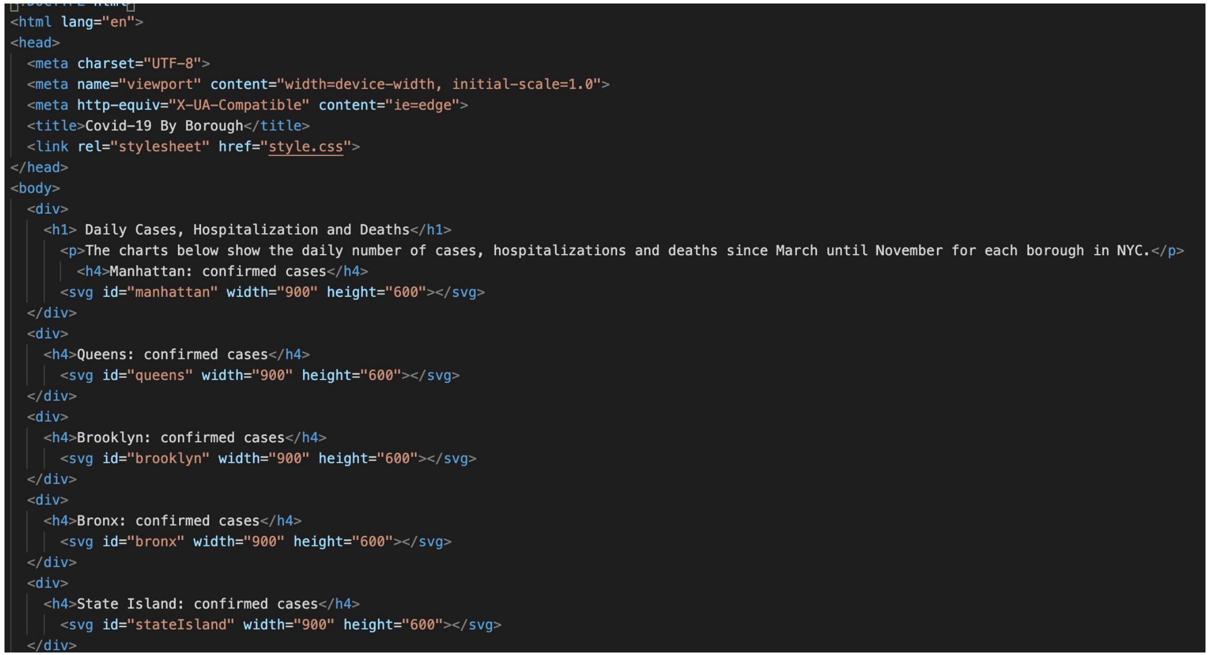
Visualizing the Pandemic

Adriana Cieslak, Robert Motta, Michael Reichard, and Luis Bejaran

The Dashboard

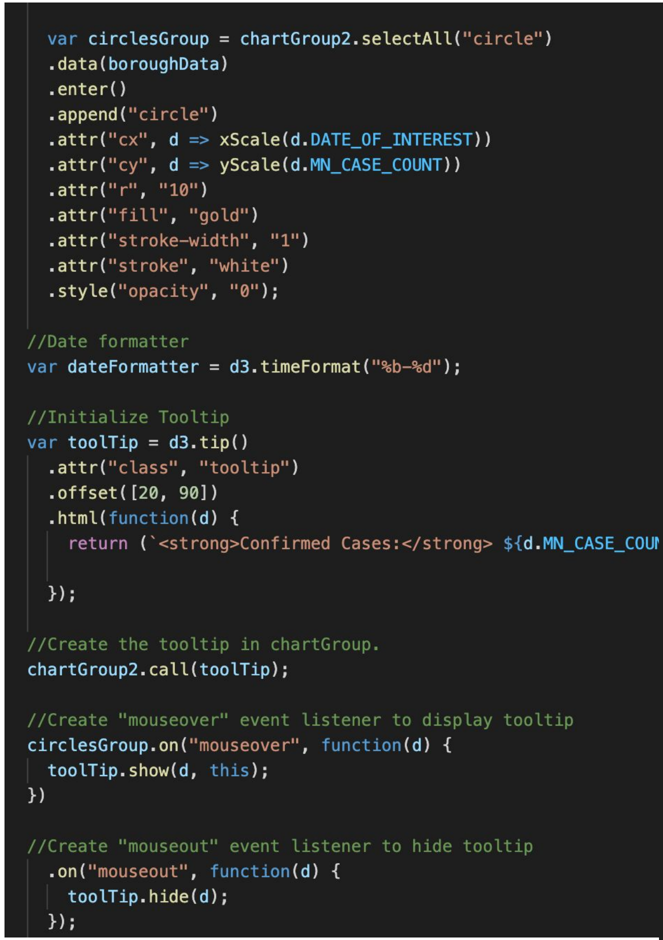
Multiple html files were used in order to have a working dashboard. From the main page users can click links that bring you to different graphs that were created by the team.



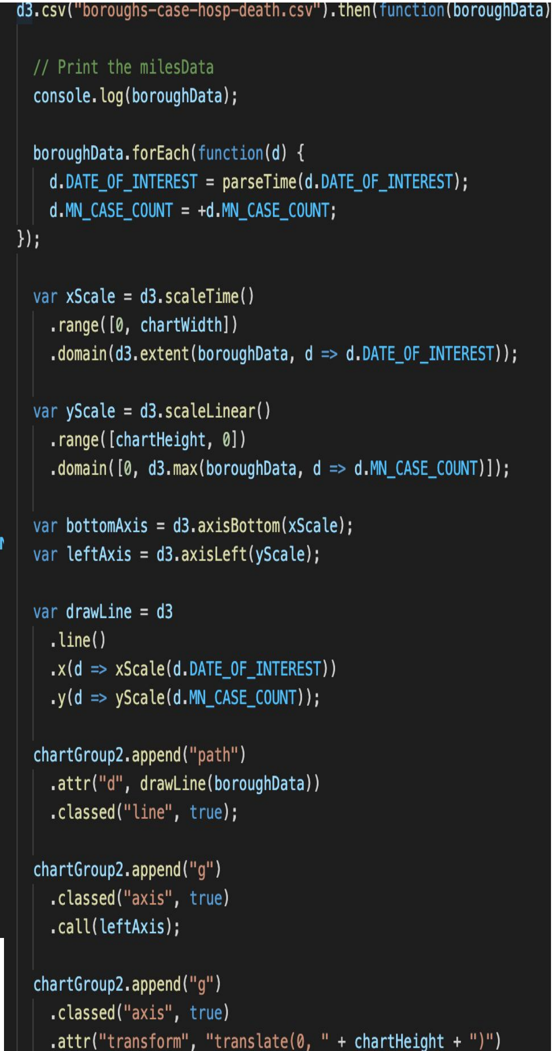
```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Covid-19 By Borough</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div>
    <h1> Daily Cases, Hospitalization and Deaths</h1>
      <p>The charts below show the daily number of cases, hospitalizations and deaths since March until November for each borough in NYC.</p>
        <h4>Manhattan: confirmed cases</h4>
      <svg id="manhattan" width="900" height="600"></svg>
  </div>
  <div>
    <h4>Queens: confirmed cases</h4>
      <svg id="queens" width="900" height="600"></svg>
  </div>
  <div>
    <h4>Brooklyn: confirmed cases</h4>
      <svg id="brooklyn" width="900" height="600"></svg>
  </div>
  <div>
    <h4>Bronx: confirmed cases</h4>
      <svg id="bronx" width="900" height="600"></svg>
  </div>
  <div>
    <h4>State Island: confirmed cases</h4>
      <svg id="stateIsland" width="900" height="600"></svg>
  </div>
```

Graphing the Raw Data

Using charts.js a graph was created to be able to view the number of new cases, hospitalizations and deaths occured on each day during the pandemic.



```
  var circlesGroup = chartGroup2.selectAll("circle")
  .data(boroughData)
  .enter()
  .append("circle")
  .attr("cx", d => xScale(d.DATE_OF_INTEREST))
  .attr("cy", d => yScale(d.MN_CASE_COUNT))
  .attr("r", "10")
  .attr("fill", "gold")
  .attr("stroke-width", "1")
  .attr("stroke", "white")
  .style("opacity", "0");

//Date formatter
var dateFormatter = d3.timeFormat("%b-%d");

//Initialize Tooltip
var toolTip = d3.tip()
  .attr("class", "tooltip")
  .offset([20, 90])
  .html(function(d) {
    return (`<strong>Confirmed Cases:</strong> ${d.MN_CASE_COUN

  });

//Create the tooltip in chartGroup.
chartGroup2.call(toolTip);

//Create "mouseover" event listener to display tooltip
circlesGroup.on("mouseover", function(d) {
  toolTip.show(d, this);
})

//Create "mouseout" event listener to hide tooltip
  .on("mouseout", function(d) {
    toolTip.hide(d);
  });
```

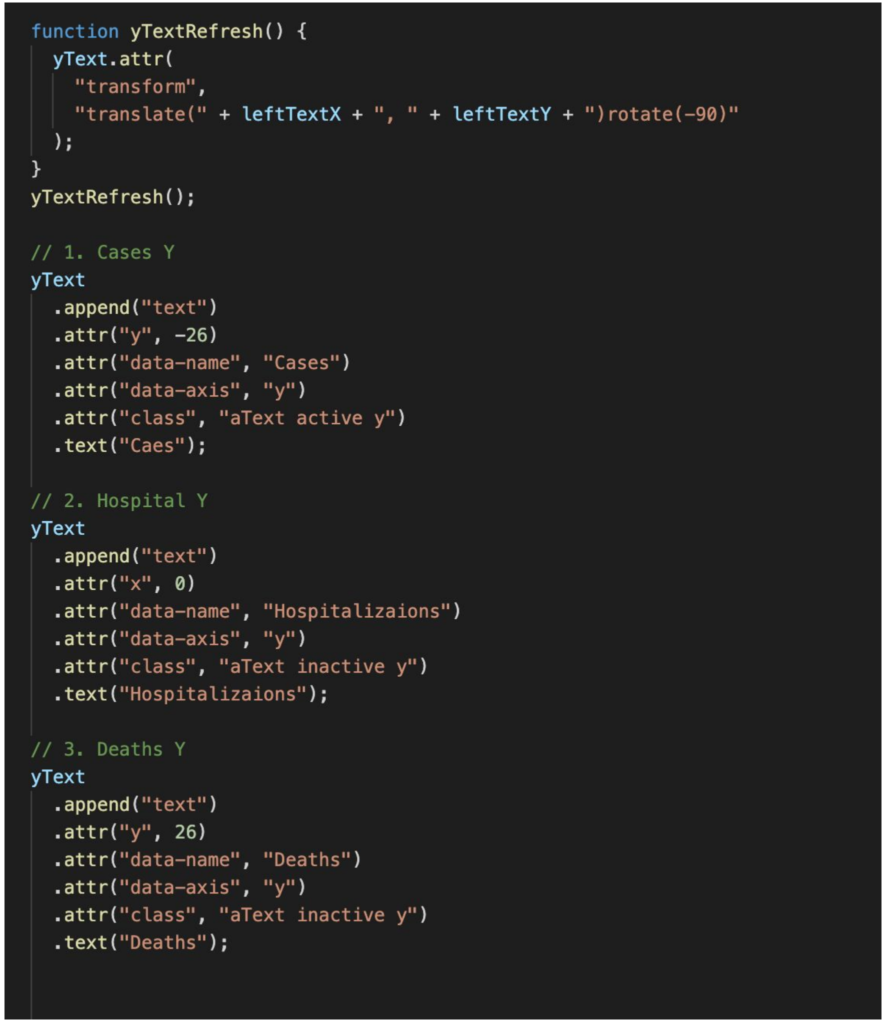


```
d3.csv("boroughs-case-hosp-death.csv").then(function(boroughData)

  // Print the milesData
  console.log(boroughData);

  boroughData.forEach(function(d) {
    d.DATE_OF_INTEREST = parseTime(d.DATE_OF_INTEREST);
    d.MN_CASE_COUNT = +d.MN_CASE_COUNT;
});

  var xScale = d3.scaleTime()
    .range([0, chartWidth])
    .domain(d3.extent(boroughData, d => d.DATE_OF_INTEREST));

  var yScale = d3.scaleLinear()
    .range([chartHeight, 0])
    .domain([0, d3.max(boroughData, d => d.MN_CASE_COUNT)]);

  var bottomAxis = d3.axisBottom(xScale);
  var leftAxis = d3.axisLeft(yScale);

  var drawLine = d3
    .line()
    .x(d => xScale(d.DATE_OF_INTEREST))
    .y(d => yScale(d.MN_CASE_COUNT));

  chartGroup2.append("path")
    .attr("d", drawLine(boroughData))
    .classed("line", true);

  chartGroup2.append("g")
    .classed("axis", true)
    .call(leftAxis);

  chartGroup2.append("g")
    .classed("axis", true)
    .attr("transform", "translate(0, " + chartHeight + ")")
```

Comparing Cases and Outcomes

Scatter Plot

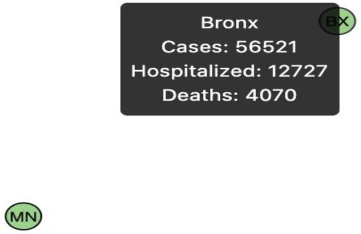
Using D3 to create a chart that allows users to see comparisons between a boroughs number of cases, hospitalizations and deaths.

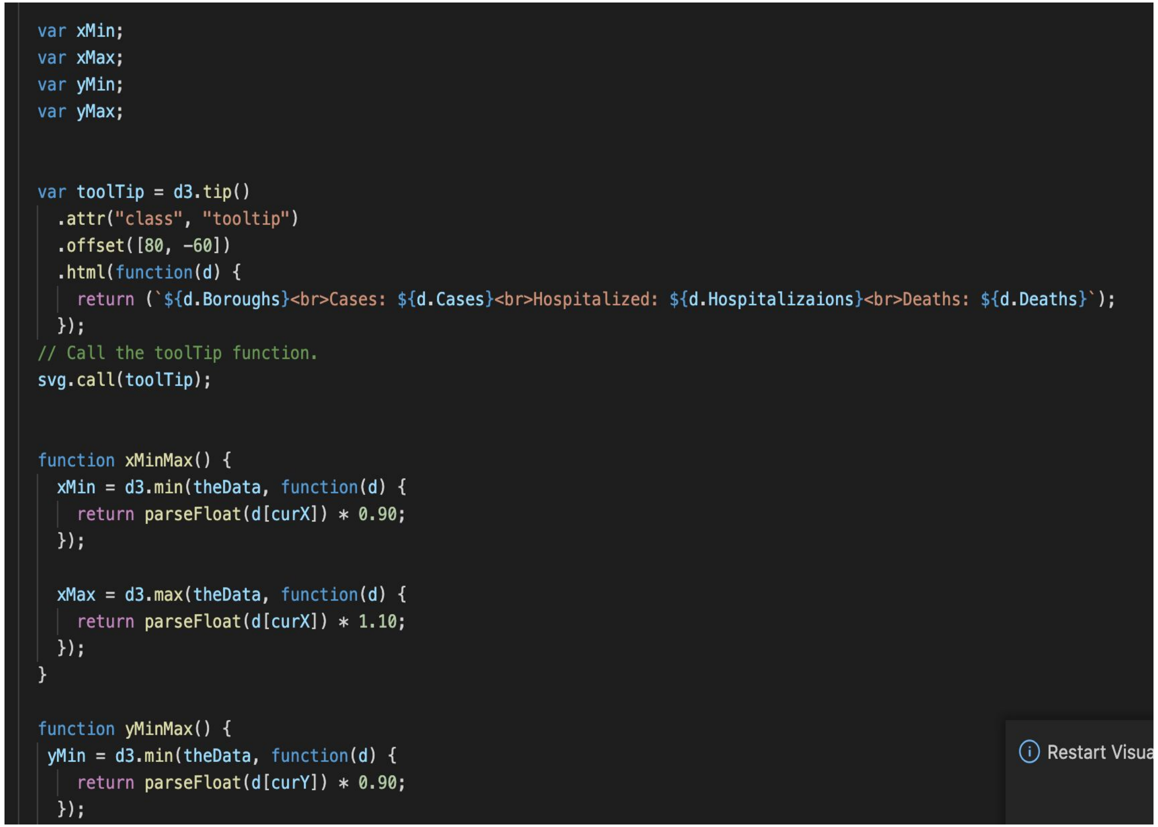


```
function yTextRefresh() {
  yText.attr(
    "transform",
    "translate(" + leftTextX + ", " + leftTextY + ")rotate(-90)"
  );
}
yTextRefresh();

// 1. Cases Y
yText
  .append("text")
  .attr("y", -26)
  .attr("data-name", "Cases")
  .attr("data-axis", "y")
  .attr("class", "aText active y")
  .text("Caes");

// 2. Hospital Y
yText
  .append("text")
  .attr("x", 0)
  .attr("data-name", "Hospitalizaions")
  .attr("data-axis", "y")
  .attr("class", "aText inactive y")
  .text("Hospitalizaions");

// 3. Deaths Y
yText
  .append("text")
  .attr("y", 26)
  .attr("data-name", "Deaths")
  .attr("data-axis", "y")
  .attr("class", "aText inactive y")
  .text("Deaths");
```

Show Data for Points

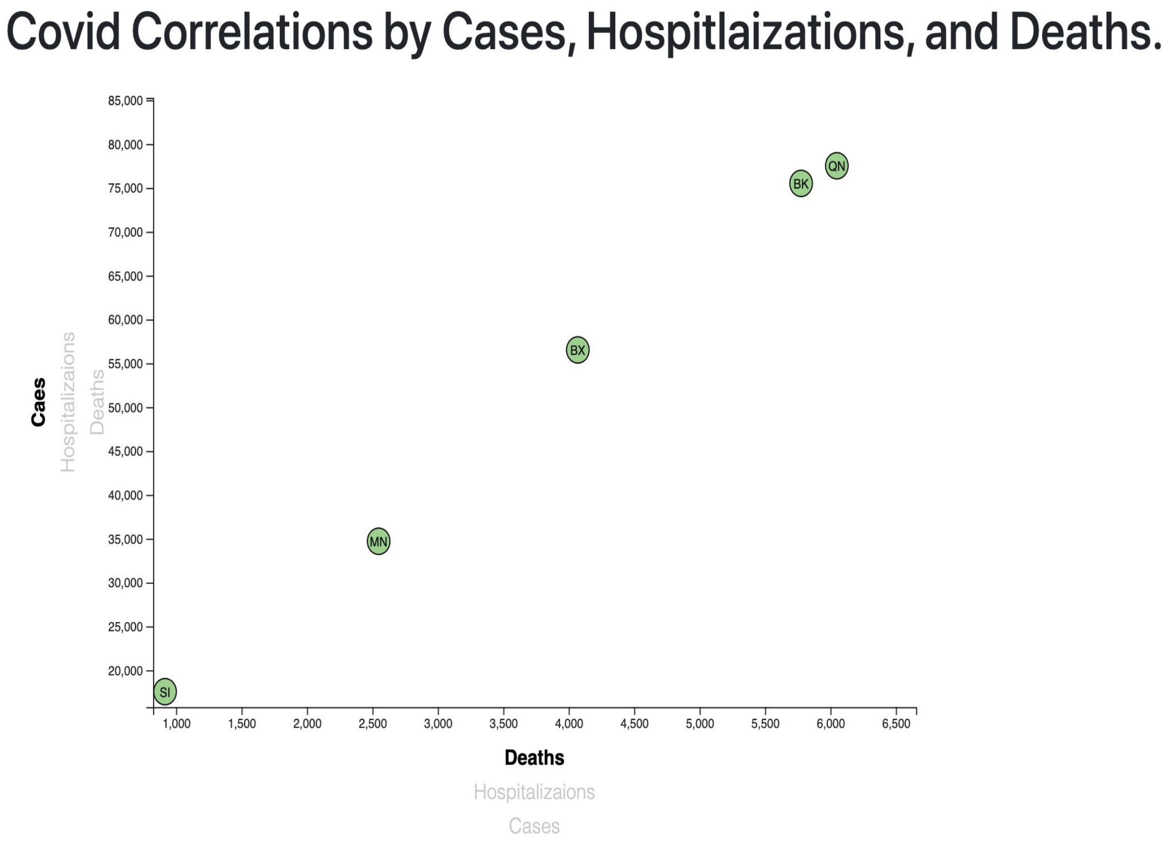
By hovering over each point on the chart users are able to view the raw data for each borough that was used to generate the chart.


Bronx
Cases: 56521
Hospitalized: 12727
Deaths: 4070


```
var xMin;
var xMax;
var yMin;
var yMax;


var toolTip = d3.tip()
  .attr("class", "tooltip")
  .offset([80, -60])
  .html(function(d) {
    return (`${d.Boroughs}<br>Cases: ${d.Cases}<br>Hospitalized: ${d.Hospitalizaions}<br>Deaths: ${d.Deaths}`);
  });
// Call the toolTip function.
svg.call(toolTip);


function xMinMax() {
  xMin = d3.min(theData, function(d) {
    return parseFloat(d[curX]) * 0.90;
  });

  xMax = d3.max(theData, function(d) {
    return parseFloat(d[curX]) * 1.10;
  });
}

function yMinMax() {
 yMin = d3.min(theData, function(d) {
    return parseFloat(d[curY]) * 0.90;
  });
```

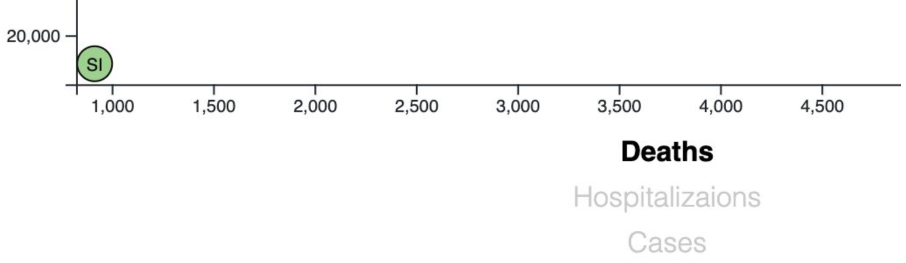
Final Visualization

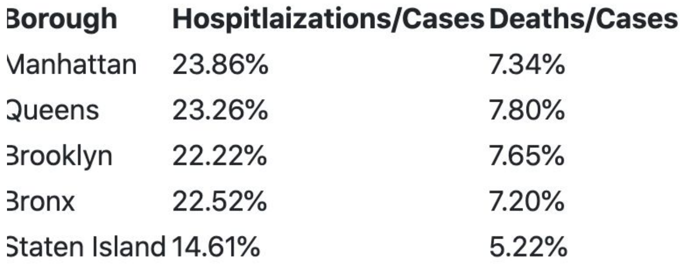
Finally the chart that was generated with all of the code!



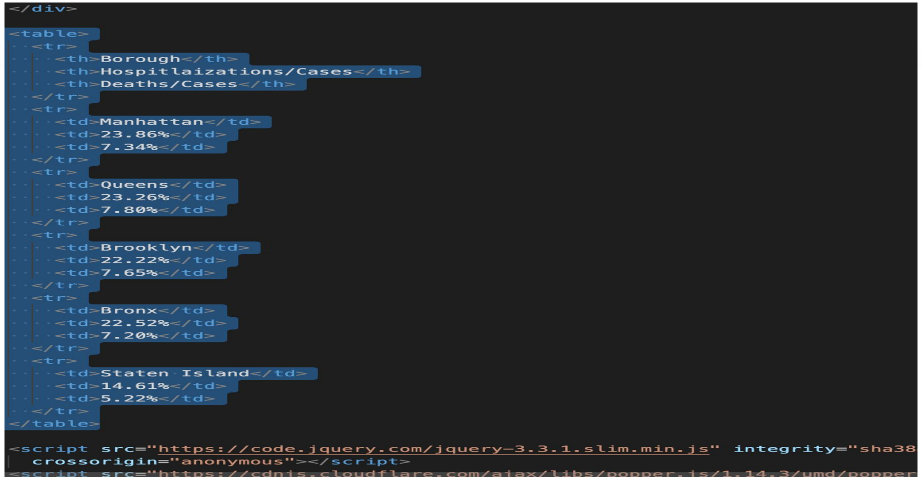
Showing Relationships

In order to make it easier for users to see how many people went to the hospital or died of those who tested positive i included a chart to be able to easily view that





Borough	Hospitlaizations/Cases	Deaths/Cases
Manhattan	23.86%	7.34%
Queens	23.26%	7.80%
Brooklyn	22.22%	7.65%
Bronx	22.52%	7.20%
Staten Island	14.61%	5.22%



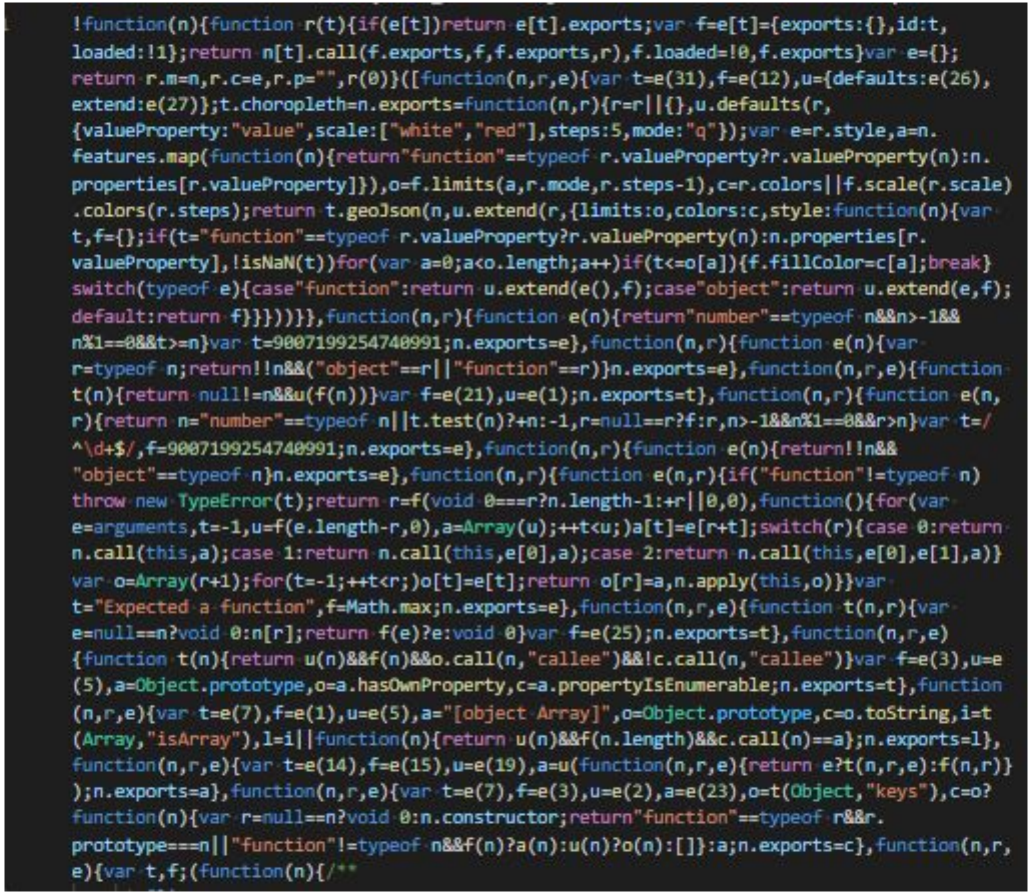
```
</div>

<table>
  <tr>
    <th>Borough</th>
    <th>Hospitlaizations/Cases</th>
    <th>Deaths/Cases</th>
  </tr>
  <tr>
    <td>Manhattan</td>
    <td>23.86%</td>
    <td>7.34%</td>
  </tr>
  <tr>
    <td>Queens</td>
    <td>23.26%</td>
    <td>7.80%</td>
  </tr>
  <tr>
    <td>Brooklyn</td>
    <td>22.22%</td>
    <td>7.65%</td>
  </tr>
  <tr>
    <td>Bronx</td>
    <td>22.52%</td>
    <td>7.20%</td>
  </tr>
  <tr>
    <td>Staten Island</td>
    <td>14.61%</td>
    <td>5.22%</td>
  </tr>
</table>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha38
  crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper
```

NYC Borough Covid Maps

Using Choropleth Shapes of NYC with Leaflet.JS to compare key metrics across the 5 Boroughs

Choropleth Shapes

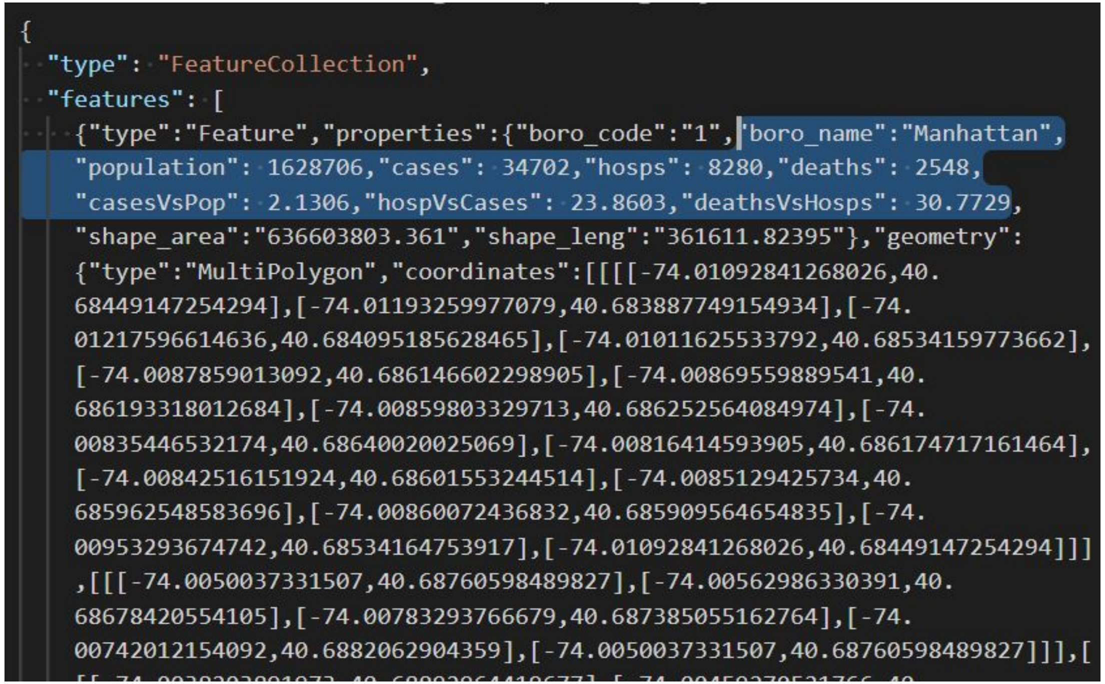
Leaflet.JS javascript file that allows you to use pre-built functions to display maps colored with Choropleth coordinates



```
!function(n){function r(t){if(e[t])return e[t].exports;var f=e[t]={exports:{},id:t,
loaded:!1};return n[t].call(f.exports,f,f.exports,r),f.loaded=!0,f.exports}var e={};
return r.m=n,r.c=e,r.p="",r(0)}([function(n,r,e){var t=e(31),f=e(12),u={defaults:e(26),
extend:e(27)};t.choropleth=n.exports=function(n,r){r=r||{},u.defaults(r,
{valueProperty:"value",scale:["white","red"],steps:5,mode:"q"});var e=r.style,a=n.
features.map(function(n){return"function"==typeof r.valueProperty?r.valueProperty(n):n.
properties[r.valueProperty]}),o=f.limits(a,r.mode,r.steps-1),c=r.colors||f.scale(r.scale)
.colors(r.steps);return t.geoJson(n,u.extend(r,{limits:o,colors:c,style:function(n){var
t,f={};if(t="function"==typeof r.valueProperty?r.valueProperty(n):n.properties[r.
valueProperty],!isNaN(t))for(var a=0;a<o.length;a++)if(t<=o[a]){f.fillColor=c[a];break}
switch(typeof e){case"function":return u.extend(e(),f);case"object":return u.extend(e,f);
default:return f}}}))}},function(n,r){function e(n){return"number"==typeof n&&n>-1&&
n%1==0&&t>=n}var t=9007199254740991;n.exports=e},function(n,r){function e(n){var
r=typeof n;return!!n&&("object"==r||"function"==r)}n.exports=e},function(n,r,e){function
t(n){return null!=n&&u(f(n))}var f=e(21),u=e(1);n.exports=t},function(n,r){function e(n,
r){return n="number"==typeof n||t.test(n)?+n:-1,r=null==r?f:r,n>-1&&n%1==0&&r>n}var t=/
^\d+$/,f=9007199254740991;n.exports=e},function(n,r){function e(n){return!!n&&
"object"==typeof n}n.exports=e},function(n,r){function e(n,r){if("function"!=typeof n)
throw new TypeError(t);return r=f(void 0===r?n.length-1:+r||0,0),function(){for(var
e=arguments,t=-1,u=f(e.length-r,0),a=Array(u);++t<u;)a[t]=e[r+t];switch(r){case 0:return
n.call(this,a);case 1:return n.call(this,e[0],a);case 2:return n.call(this,e[0],e[1],a)}
var o=Array(r+1);for(t=-1;++t<r;)o[t]=e[t];return o[r]=a,n.apply(this,o)}}var
t="Expected a function",f=Math.max;n.exports=e},function(n,r,e){function t(n,r){var
e=null==n?void 0:n[r];return f(e)?e:void 0}var f=e(25);n.exports=t},function(n,r,e)
{function t(n){return u(n)&&f(n)&&o.call(n,"callee")&&!c.call(n,"callee")}var f=e(3),u=e
(5),a=Object.prototype,o=a.hasOwnProperty,c=a.propertyIsEnumerable;n.exports=t},function
(n,r,e){var t=e(7),f=e(1),u=e(5),a="[object Array]",o=Object.prototype,c=o.toString,i=t
(Array,"isArray"),l=i||function(n){return u(n)&&f(n.length)&&c.call(n)==a};n.exports=l},
function(n,r,e){var t=e(14),f=e(15),u=e(19),a=u(function(n,r,e){return e?t(n,r,e):f(n,r)}
);n.exports=a},function(n,r,e){var t=e(7),f=e(3),u=e(2),a=e(23),o=t(Object,"keys"),c=o?
function(n){var r=null==n?void 0:n.constructor;return"function"==typeof r&&r.
prototype===n||"function"!=typeof n&&f(n)?a(n):u(n)?o(n):[]}:a;n.exports=c},function(n,r,
e){var t,f;(function(n){/**
```

GeoJSON

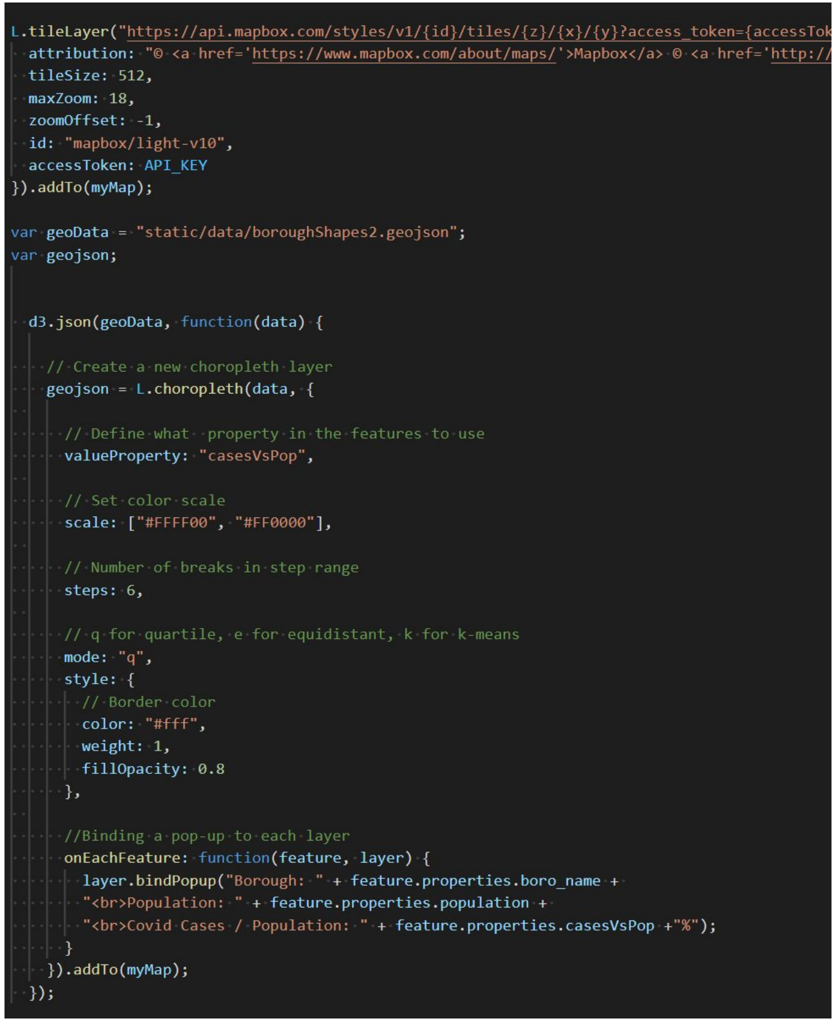
By editing a basic a geoJSON file, mapping the choropleth shapes of NYC boroughs, we were more easily able to manipulate Covid data, using the Leaflet.JS library.



```
{
  "type": "FeatureCollection",
  "features": [
    {"type":"Feature","properties":{"boro_code":"1","boro_name":"Manhattan",
    "population": 1628706,"cases": 34702,"hosps": 8280,"deaths": 2548,
    "casesVsPop": 2.1306,"hospVsCases": 23.8603,"deathsVsHosps": 30.7729,
    "shape_area":"636603803.361","shape_leng":"361611.82395"},"geometry":
    {"type":"MultiPolygon","coordinates":[[[[-74.01092841268026,40.
    68449147254294],[-74.01193259977079,40.683887749154934],[-74.
    01217596614636,40.684095185628465],[-74.01011625533792,40.68534159773662],
    [-74.0087859013092,40.686146602298905],[-74.00869559889541,40.
    686193318012684],[-74.00859803329713,40.686252564084974],[-74.
    00835446532174,40.68640020025069],[-74.00816414593905,40.686174717161464],
    [-74.00842516151924,40.68601553244514],[-74.0085129425734,40.
    685962548583696],[-74.00860072436832,40.685909564654835],[-74.
    00953293674742,40.68534164753917],[-74.01092841268026,40.68449147254294]]]
    ,[[[-74.0050037331507,40.68760598489827],[-74.00562986330391,40.
    68678420554105],[-74.00783293766679,40.687385055162764],[-74.
    00742012154092,40.6882062904359],[-74.0050037331507,40.68760598489827]]],[
```

Mapbox/D3/Leaflet

Ultimately, we created a the map layer by calling the Mapbox api, then unpacked the geojson data containing choropleth shapes and Covid data, to finally use leaflet features to create the overall display and interactivity of the maps.



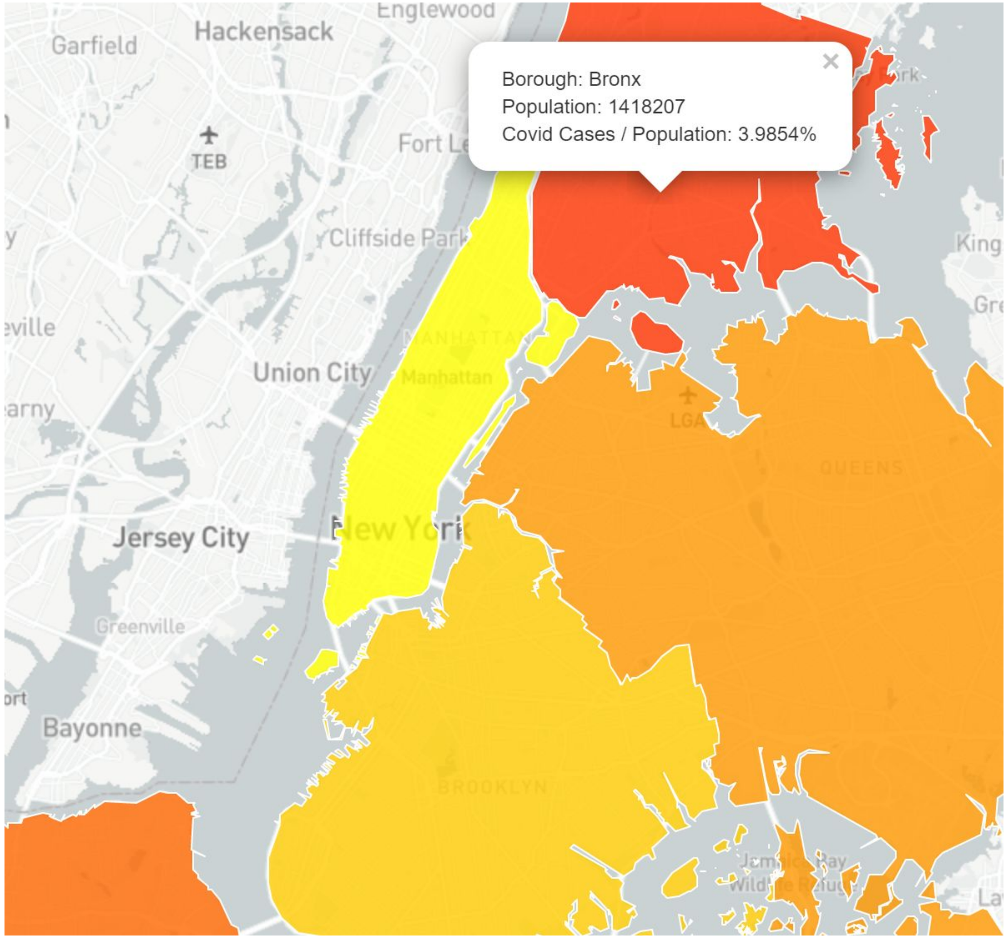
```
L.tileLayer("https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessTok
  attribution: "© <a href='https://www.mapbox.com/about/maps/'>Mapbox</a> © <a href='http://
  tileSize: 512,
  maxZoom: 18,
  zoomOffset: -1,
  id: "mapbox/light-v10",
  accessToken: API_KEY
}).addTo(myMap);

var geoData = "static/data/boroughShapes2.geojson";
var geojson;


  d3.json(geoData, function(data) {

    // Create a new choropleth layer
    geojson = L.choropleth(data, {

      // Define what  property in the features to use
      valueProperty: "casesVsPop",

      // Set color scale
      scale: ["#FFFF00", "#FF0000"],

      // Number of breaks in step range
      steps: 6,

      // q for quartile, e for equidistant, k for k-means
      mode: "q",
      style: {
        // Border color
        color: "#fff",
        weight: 1,
        fillOpacity: 0.8
      },

      //Binding a pop-up to each layer
      onEachFeature: function(feature, layer) {
        layer.bindPopup("Borough: " + feature.properties.boro_name +
        "<br>Population: " + feature.properties.population +
        "<br>Covid Cases / Population: " + feature.properties.casesVsPop +"%");
      }
    }).addTo(myMap);
  });
```

Maps

Using 3 data rates by Borough
- Cases vs Population
- Hospitalizations vs Cases
- Deaths vs Hospitalizations

We created maps where the boroughs were shaded from yellow to red based on the severity of these rates.

Clicking on any borough will pop-up a summary of that data point for that borough


Borough: Bronx
Population: 1418207
Covid Cases / Population: 3.9854%

What I would have liked to improve/add

- My GEOJson was manually edited to add covid rates. I would have liked to append that data more safely with code, like jsonify
- Maps that could be scrolled through with a control by date, so you could track color changes as pandemic progressed