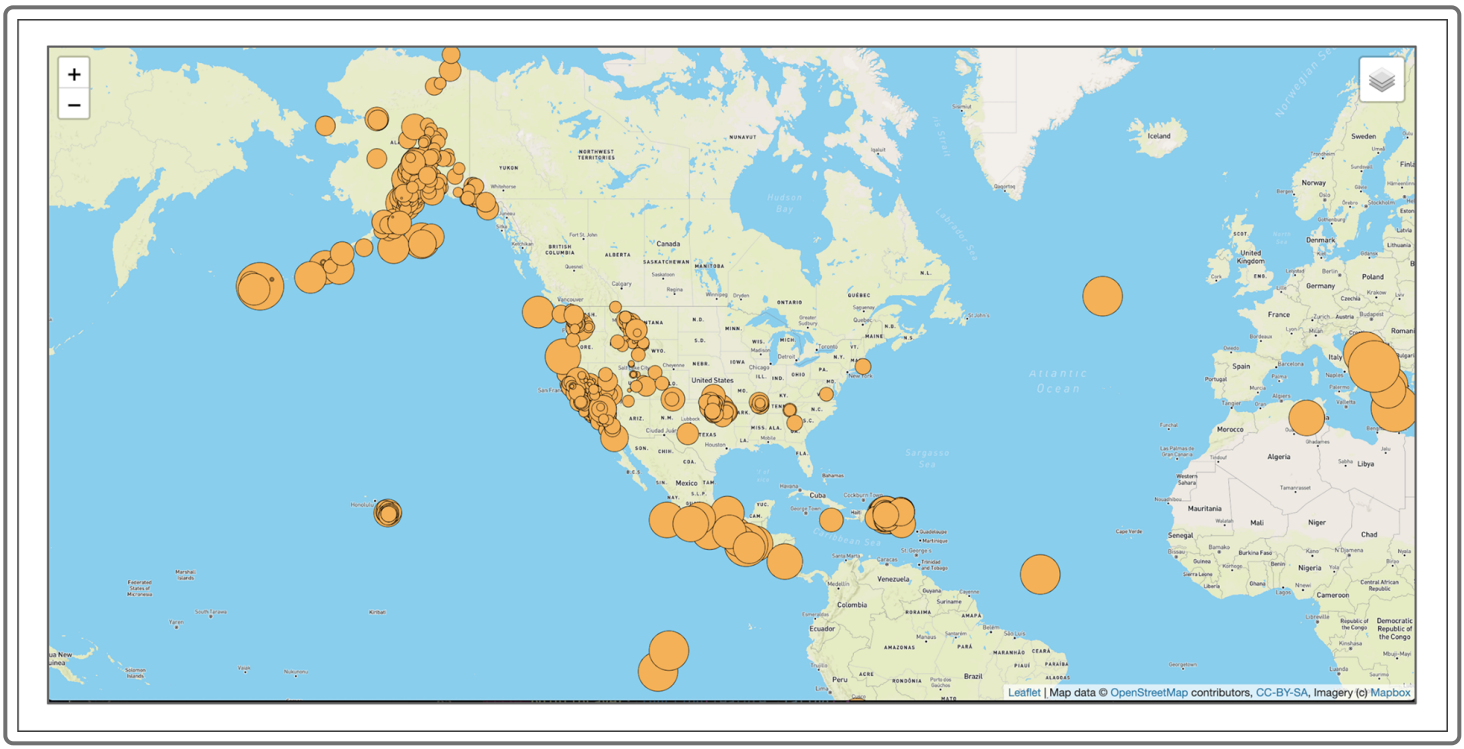
13.6.2

Add Style to the Earthquake Data

As a first step in making the earthquake data more visually appealing, Sadhana would like you to add some styling to the earthquake data and vary the radius of each earthquake based on the magnitude.

After styling and modifying the radius of the circle for each earthquake's magnitude, our map should look similar to the following map:



Before we write the code to create this map, make a copy of the `logicStep1.js` file and name it `logicStep2.js`. Now let's edit the file.

First, we'll change the basic marker to a circleMarker by using the `pointToLayer` function.

REWIND

For the `pointToLayer` callback function, the basic syntax for adding functionality to a marker is:

```
L.geoJSON(data, {
pointToLayer: function(feature, latlng) {
return L.marker(latlng);
}
});
```

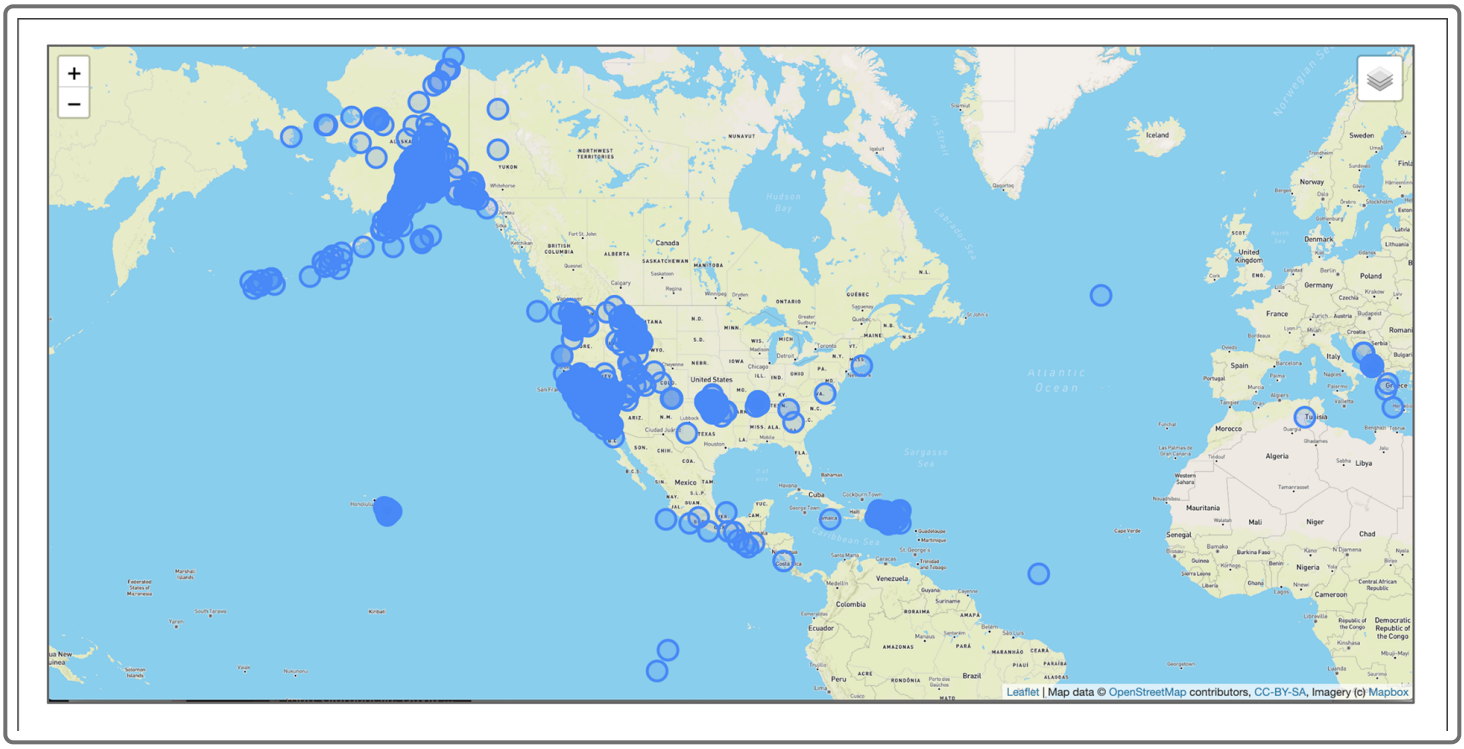
For our purposes, we'll use `circleMarker` instead of `marker` in the above code. Edit your GeoJSON layer code to look like the following:

```
// Creating a GeoJSON layer with the retrieved data.
  L.geoJSON(data, {

// We turn each feature into a circleMarker on the map.

pointToLayer: function(feature, latlng) {
            console.log(data);
            return L.circleMarker(latlng);
        },
    }).addTo(map);
});
```

Save the file and let's see what the data looks like on the map. The `index.html` file should look like the following:



Next, we'll create a style for each earthquake by adjusting the line color, fill color, opacity, fill opacity, stroke, weight, and radius.

REWIND

When we defined the line style for the nonstop flight routes from Toronto, we created a style variable like the following:

```
let myStyle = {
color: "#ffffa1",
weight: 2
}
```

We'll create a `function styleInfo()`, which will contain all the style parameters for each earthquake plotted. Within this function, we'll create a `getRadius()` function to calculate the radius for each earthquake.

Add the following `function styleInfo()` inside the `d3.json()` method:

```
// This function returns the style data for each of the earthquakes we plot on
// the map. We pass the magnitude of the earthquake into a function
// to calculate the radius.
function styleInfo(feature) {
  return {
    opacity: 1,
    fillOpacity: 1,
    fillColor: "#ffae42",
    color: "#000000",
    radius: getRadius(),
    stroke: true,
    weight: 0.5
  };
}
```

Let's review the style we're creating for each earthquake:

- In the `styleInfo()` function, we passed the argument `feature` to reference each object's features.
- The `opacity` and `fillOpacity` are set at 1, the `stroke` is "true," and the `weight` is 0.5.
- The `fillColor` is light orange, and the color is `"#000000"` (black).
- The `getRadius()` function retrieves the earthquake's magnitude. Next, we'll create the `getRadius()` function to calculate the radius of the circle from the magnitude.

Retake

In the `getRadius()` function for our `styleInfo()` function, add the following code to retrieve the earthquake's magnitude: `feature.properties.mag`.

Next, we'll create the `getRadius()` function. Add the following code below the `styleInfo()` function:

```
// This function determines the radius of the earthquake marker based on its magnitude.
// Earthquakes with a magnitude of 0 will be plotted with a radius of 1.
```

```
function getRadius(magnitude) {
  if (magnitude === 0) {
    return 1;
  }
  return magnitude * 4;
}
```

In the `getRadius()` function, we'll pass the `magnitude` argument that will reference the `feature.properties.mag` in the `styleInfo()` function. Then we'll use a conditional statement that sets the magnitude to 1 if the magnitude of the earthquake in the JSON file is 0 so that the earthquake is plotted on the map. If the magnitude is greater than 0, then the magnitude is multiplied by 4.

Now, that we created our style, let's add it to the map.

Retake

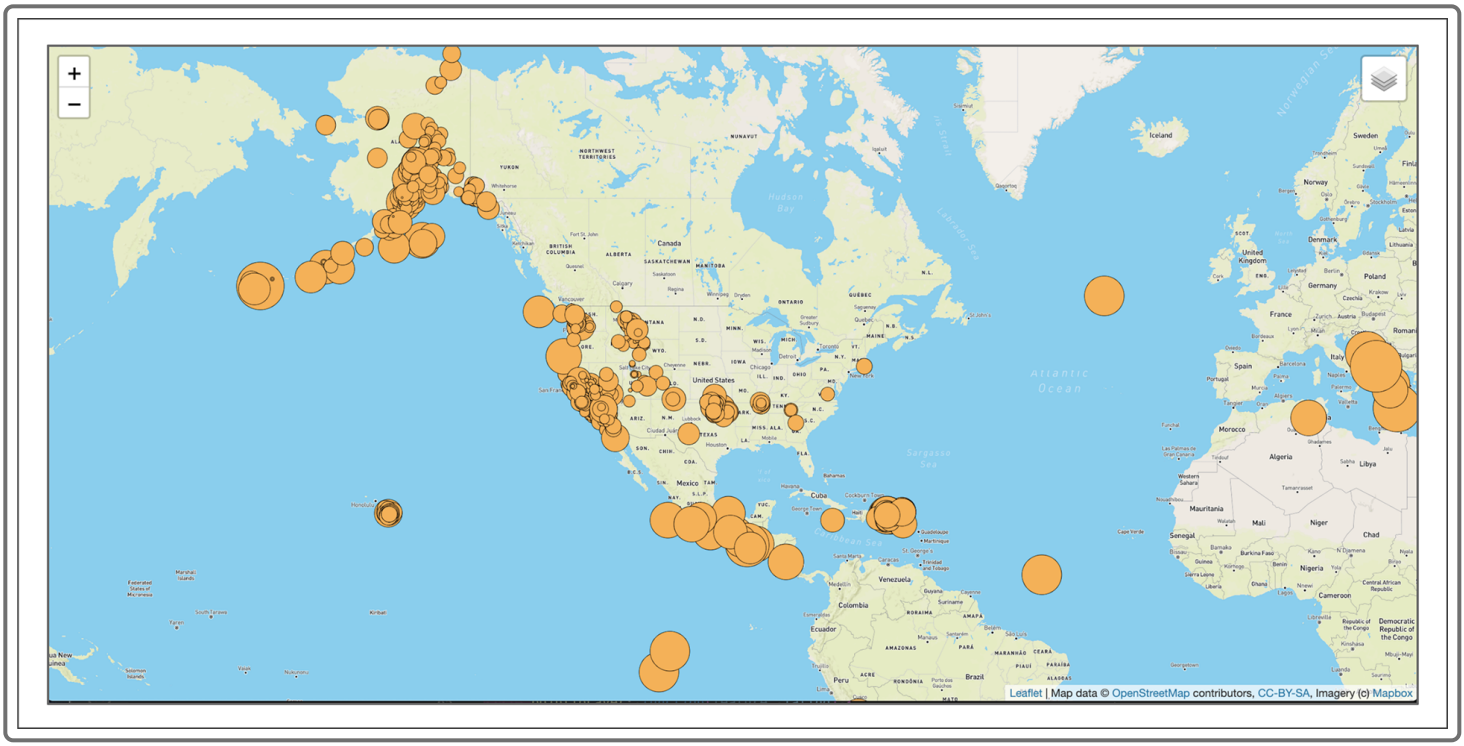
To add style to the `L.geoJSON()` layer, the `style` key will be assigned to the `styleInfo` function we created. Make sure the code for your `L.geoJSON()` layer looks like the following:

```
// Creating a GeoJSON layer with the retrieved data.
  L.geoJSON(data, {

// We turn each feature into a circleMarker on the map.

pointToLayer: function(feature, latlng) {
            console.log(data);
            return L.circleMarker(latlng);
        },
      // We set the style for each circleMarker using our styleInfo function.
    style: styleInfo
    }).addTo(map);
});
```

When you save your `logicStep2.js` file and open `index.html` in your browser, your map will look like the following:



Great job styling each earthquake on our map!

ADD/COMMIT/PUSH

Add, commit, and push your changes to your Earthquakes_past7days branch.

Let's continue making the earthquake data visually appealing by styling colors to represent magnitudes as well as by adding informational popups.