

Geography 825

Lab Seven ALTERNATE: Geoviz / Animation

This lab introduces animation functions to basic Leaflet base maps. This information is provided via third-party plugins, which are commonly used, independently developed components of web-based geovisualizations. The plugin is called Leaflet Timeline (or `timeline.leaflet`), and – by default - it looks for the “start” and “end” time fields with attributes in full Unix time, a 13-digit unit that represents the number of milliseconds since the beginning of New Year’s Day on January 1, 1970 at midnight UTC. For example, this exercise becomes available at 1603978200000. We will work with 2020 polling data to create animated, interactive web maps.

Begin working from the template

- Open your **text editor** (Notepad++ or Atom or other) and begin by opening the HTML file provided – `lab9part1.html`. Or, copy from the example text:

```
<html>
  <head>
    <title>Leaflet.timeline Example - Country Borders</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"/>
    <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
    <script src="https://skeate.dev/Leaflet.timeline/examples/leaflet.timeline.js"></script>

    <style>
      html,
      body {
        margin: 0;
        padding: 0;
      }
      #map {
        width: 100vw;
        height: 100%;
      }
      .leaflet-bottom.leaflet-left {
        width: 100%;
      }
      .leaflet-control-container .leaflet-timeline-controls {
        box-sizing: border-box;
        width: 100%;
        margin: 0;
        margin-bottom: 25px;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
```

```

<script>
    function getColorFor(value) {
        return value >= 1 ? '#2166ac':
            value >= 0 ? '#ffffbf':
                '#b2182b';
    }

    var background = L.tilelayer('https://stamen-tiles-{s}.a.ssl.fastly.net/toner-
background/{z}/{x}/{y}{r}.{ext}', {
        attribution: '',
        subdomains: 'abcd',
        minZoom: 0,
        maxZoom: 20,
        ext: 'png'
    });

    var map = L.map("map", {
        layers: [background],
        center: new L.LatLng(38, -95),
        zoom: 5,
    });
    var timeline;
    var timelineControl;

    function onLoadData(data) {
        timeline = L.timeline(data, {
            style: function (data) {
                return {
                    color: getColorFor(data.properties.DIFF),
                    fillOpacity: 0.9,
                };
            },
            waitToUpdateMap: true,
            onEachFeature: function (feature, layer) {
                layer.bindTooltip('<h3>' + feature.properties.NAME + '</h3> <p>' + '
<br>Biden: <strong>' + feature.properties.DEM + '</strong> <br>Trump: <strong>' +
feature.properties.REP + '</strong></p>');
            },
        });
    });

    timelineControl = L.timelineSliderControl({
        formatOutput: function (date) {
            return new Date(date).toLocaleDateString();
        },
        enableKeyboardControls: true,
    });
    timeline.addTo(map);
    timelineControl.addTo(map);
    timelineControl.addTimelines(timeline);
}
</script>
<script src="https://electionmaps.github.io/statepolls5.jsonp"></script>
</body>
</html>

```

- This creates a timeline-enabled map and sets it to a GeoJSON type file that I'm hosting on my site. I've provided you with a copy of this GeoJSON file in the Lab 7a folder (statepolls825.geojson). You might recall from earlier classes that you can bring in GeoJSON files as regular JSON, but you probably have not seen "padded" json, **.jsonp** files. This just helps protect your data from cross-domain issues that might develop (for example, if you are bringing in files from https:// and http:// or locally, you may run into issues). The only thing you will note if you scrutinize the hosted file is that I have changed it to put the whole GeoJSON file in parentheses preceded by "onLoadData." This is a function I use in the HTML document around line 54, and it just tells the browser this is the dataset to load in. *If and when you replace the GeoJSON, you will want to do the same thing, onLoadData({geojson goes here}) and save as a .jsonp file.*
- Also in the onLoadData function around line 54, I've already set things up to colorize states based on whether they are trending toward Joe Biden or Donald Trump, based on opinion polls. **Question 1: What data field in our layer is being used for color info and how do you know?** _____ (note here I'm not asking about the breaks or the specific colors, just what attribute field is being used; I explain styling like breaks and colors just below)
- You see the breaks are being set in the "getColorFor" function higher up, near line 33. This function says to look for numerical values (value) in the data to which you point it, and then classify them based on some criteria you also set:


```
function getColorFor(value) {
    return value >= 1 ? '#2166ac':
           value >= 0 ? '#ffffbf':
           '#b2182b';
}
```

- The function to set colors here assigns a HEX color code to various arguments. Here, if the difference field (DIFF) is greater than 1, meaning Biden outpolled Trump by 1 percent or more (greater than or equal to 1), it colors it dark blue; if there is no difference, then it assigns the shape to yellow; and if none of these criteria are met, it is colored dark red. (*Notably, this only works because data are integers, so any ties are 0 and any leads are at least 1*) You could add as many arguments as you would like in this fashion. For

example, you could say that values greater than or equal to -5 should be light red instead, by inserting that line after the third line, “value >= 0...”

You would write: **value >= -5 ? '#add8e6'**: to do this

- Essentially what this says is if the value is greater than or equal to -5, then assign this light red color, otherwise...” That colon allows for the next “if” argument by introducing the “else” to the end argument. And typically when using greater than, you’d layer on higher values on top and lower values below, because that way it will work through the data and work its way through until it meets a set of criteria. The final argument is what it will do if none of the values are met. In the example HTML, I closed with the simple argument **‘#b2182b’** ; which says to make the shape dark red, and then end the function. This only fires if all other criteria are false, so this is typically the last color you’d assign in a choropleth. In the example, it’s applied to everything where the difference in support between Biden and Trump is negative, indicating anywhere that Trump holds a lead in opinion polls.
- **Question 2: How would you rewrite the top part of the getColorFor function so that there is an intermediate, light-blue tier with a break at 3?**
- Now, go through and create at least 7 classes, in a diverging scheme, for the choropleth. I recommend using Color Brewer for identifying hex colors and schemes that would work.
- Change the background layer if desired.
- When ready, create a Lab 7a folder and upload your file to your GitHub pages repository. In your answers document, provide a link to your finished site.
- **Question 3: Reflect briefly on what elements work well here and which might confuse audiences. How would you improve this map?**
Next, you will create a second map using the same underlying data, but you will turn it into a proportional map for use with the slide. It involves some more work on your end, and you’ll have to jump into a desktop GIS software to complete it.
- Open the GeoJSON file I provided (statepolls825.geojson) in QGIS.

- Open the attribute table and notice that you do not have a field for electoral votes.
- Add the EV.CSV file to QGIS. Click the radio button for “No geometry.”
- Then double-click the statepolls layer. Go to the joins panel  and join the EV.CSV to the layer based on the state name fields.
- Export the joined GeoJSON layer to a new GeoJSON file. EDIT this file so that Nebraska’s second district has 1 electoral vote
- Next, run the “Centroids” tool (vector geometry) on the new layer. Save the new layer as a geojson.
- Edit the new GeoJSON file by moving Alaska and Hawaii closer to the contiguous states. [Consult the QGIS Documentation](#) if you have forgotten how to edit features (or just move it all to ArcMap if you can’t stand it?). Then save, or export to a new file.
- Open your new GeoJSON file in a text editor. Make sure to copy the example and add “onLoadData(“ to the beginning of the file and then scroll to the very bottom and add the final “)” making sure that the entire GeoJSON file is contained in the onLoadData() function. Save it as a .jsonp format. Upload your new file to your GitHub Pages account and make sure to remember the URL, because you’ll use it in a minute.
- Return to your text editor and open the second HTML file I provided (lab9part2.html) and save it with a new name.
- Replace the script source at the bottom with the link to your .jsonp file
- Notice – near line 98 – that radius is used to scale symbols, and in this template I have it set to the electoral vote value field.
- **Question 4: Briefly, why is scaling radius by a direct value bad for symbol mapping? Will resulting symbols be proportional?**

- Now, we will modify the radius field so that it proportionally scales symbols based on incoming data somewhat better
- First, we set the lowest symbol sizes/values we want, then create a function to calculate symbol sizes, and finally, we will use the radius field to leverage that function.
- Insert two variables into the code near the top, just above your tile layer

```
var minValue = 1;  
var minRadius = 4;
```

- Next, we'll write a function for scaling up from that minimum point. Stick this just below the variables we just added -

```
function calcRadius(val) {  
  return 1.0083 * Math.pow(val/minValue,.5716)*minRadius;  
}
```

- Finally, change your radius definition line, which is probably just below line 100 now, from "radius: data.properties.EV," to "calcRadius(data.properties.EV),"

This tells it to run the calculation on the EV field in your dataset.

- Now, just adjust the symbol scaling as you see fit. Overlap is okay – and expected – enabled by transparency in this case.
- Upload your completed file to your GitHub Pages repository and share the URL with me.
- Submit your answers and links in an attached text document!