Lab 6 Additional Exercise!

So while this choropleth map looks okay, it doesn't actually do very much to help out the audience, even if we added a legend. That's because there are too many values in the mid-range: this particular part of Lincoln saw a lot of development in the 1950s.

Sometimes even when you pick the right choropleth shading, the best map projection for your area, et cetera… you still fail to deliver an interactive map that sufficiently communicates a subject or topic to an audience. Maps are supposed to tell a story right away – we discussed this very basic but important concept early on in Cartography I. With web cartography, we often expose our data and maps to additional layers of interactivity, which might indicate more confidence in our subject or at least leave some audiences feeling like their takeaways are even more valid. But this doesn't have to be a bad thing. In moderation, we can arm our viewers with even more exploratory tools, taking seriously our responsibility to get things "right."

So what's a good supporting element here? There are many options, but for the sake of time, let's use a time slider (heh heh). But *this* time (I'm on fire now), unlike in previous labs, we're not going to be making some large code block that parses through a spreadsheet and maps out coordinates and draws circles at them differently. Instead, we will actually just write a very small code block and make a couple of minor changes so that we can pull out an underlying field you may have noticed in the JSON: a date field. Then we will write a tiny little code block that matches that date field up with a value in the slider and *voila*!

This is an extra credit assignment, technically, but it's fast and if you've just done Lab 6, it is worth the effort (I think [otherwise I wouldn't have spent all this time writing it]).

FIRST.

Return to your Lab 6 HTML. Let's give it a new name. "Lab6_2.html," "Lab6p2.html," I think you can just decide what you like best. Some of you probably will, as usual, write some really weird name for this, and that's fine.

- We will need to first add just a tiny section to our style/css that indicates where we want to put a slider in. Put this inside the <style> tags at the top, immediately below any other style code you have added, but before the closing </style> tag:

```
        #sliderContainer {
            text-align: center;
            position: right;
            top: 700px;
        }
```

- Then, up at the top of our page, below the H2 section you like, but before the script begins, you need to throw in a “DIV” to physically house the slider on the page (Note: there are *many* other sliders available as .js code, by the way, we’re just using a super simple HTML 5 one to keep this easy)

- To do this, you just create a container at the top of the BODY but above the script (note I’m calling it “sliderContainer” so that it matches the style code exactly) and you say you want to use a range there. Here, we have about 130 years of data so I’m going to make a slider that’s about 130 “steps” long. And then I’ll tell it what year to start at, 1890:

```
<div id="sliderContainer">
  <input id="timeslide" type="range" min="0" max="129" value="0"
step="1"/><br>
        <span id="range">1890</span>
```

“But Professor Shepard, why are you using 0 and 129 and not just actual… years?”

Great question, observant student who may or may not have actually just asked that but we’ll maybe never actually know because of COVID-19.

In these examples, I will purposefully create an *adaptable* slider that lets you fill in all kinds of stuff depending on your needs (for those of you eyeing possible code examples for final projects). With this slider format, you could (for example) set the range from 0 to 11 (12 “steps) and then write the whole thing to look at months rather than years. Or days. Or different arbitrary

categories like “the before time” and “our inevitable dystopian future.” By doing it all in this manner ^^ you are only telling the slider how many stops it can have along the way. THEN, later on, you can use a variable to define what goes into those spots. Does that make sense? I hope so. If no, call me up on my Zoom meeting during office hours or class. But let’s assume that made sense and move on…

- So let’s do that thing where we identify what will populate the slider values. I’m going to save you a lot of typing and just give you all the years. Put this as a variable at the top or near the top of your actual script where you’ve defined other variables like width and height (and yes, I just saved you a lot of typing if you copy+paste properly):

```
var year =
["1890","1891","1892","1893","1894","1895","1896","1897","1898","1899","1
900","1901","1902","1903","1904","1905","1906","1907","1908","1909","1910
","1911","1912","1913","1914","1915","1916","1917","1918","1919","1920","
1921","1922","1923","1924","1925","1926","1927","1928","1929","1930","193
1","1932","1933","1934","1935","1936","1937","1938","1939","1940","1941",
"1942","1943","1944","1945","1946","1947","1948","1949","1950","1951","19
52","1953","1954","1955","1956","1957","1958","1959","1960","1961","1962"
,"1963","1964","1965","1966","1967","1968","1969","1970","1971","1972","1
973","1974","1975","1976","1977","1978","1979","1980","1981","1982","1983
","1984","1985","1986","1987","1988","1989","1990","1991","1992","1993","
1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","200
4","2005","2006","2007","2008","2009","2010","2011","2012","2013","2014",
"2015","2016","2017","2018","2019","2020"];
```

- *Right below that variable*, add another one called inputValue. I want this to be created but set to nothing - NULL. We’ll set this dynamically later on (kind of like how we left the H2 section blank in the normal Lab 6 assignment to be populated by data later on):

```
  var inputValue = null;
```

- Now, if you were carefully looking at the code a couple of pages above, you might noticed that we used “timeslide” as our time slider’s “input ID.” So here, we have a little function that tells D3 to listen for a change to the timeslide element, and then give the value selected to a function called “update.” This update function will also update and store the inputValue from our list of values when the slider changes. Here we have set fill to “dateMatch,” which is another function we’ll write that actually checks to see if the inputValue matches a building data. As you follow through the dateMatch function, you’ll see that we grab the date from a field in our buildings data that we used earlier, RESYRBLT, and since it’s a number, I convert it to a string and check if it matches the inputValue from the slider (which we know is a string since we wrote it ourselves… well at least I did… e.g. “1950”,”1951”, etc.). If in fact it does match, we say we want to color the objects yellow (if they match), otherwise leave them all gray. Put this section at the bottom of your script, but before the /Script end tag:

```
d3.select("#timeslide").on("input", function() {
        update(+this.value);
    });

    function update(value) {
        document.getElementById("range").innerHTML=year[value];
        inputValue = year[value];
        d3.selectAll(".buildingdata")
            .attr("fill", dateMatch);
    }

    function dateMatch(data, value) {
      //  var d = new Date(data.properties.TIME);
      //  var m = year[d.getFullYear()];
        var yy = data.properties.RESYRBLT;
        var y = yy.toString();
        if (inputValue == y) {
            this.parentElement.appendChild(this);
            return "yellow";
        } else {
            return "#999";
        };
    }
```

But Professor Shepard… what about all this commented out stuff? You might be asking this right now.

So, what I'm doing here is showing you ANOTHER way. That buildings data I uploaded also contains a field called TIME, which has dates written out like "November 11, 2011." In JavaScript, it is actually possible to ingest those sorts of strings and process them as dates and spit out a value. What I'm doing there is pulling in the longer date field, TIME, and I'm telling it to get the year "getFullYear()" out of it, e.g. 2011 so that we can check it against the "inputValue." There are also things like "getMonth" which YOU might find useful. So I left those commented (but not active) so that you could comment out the way I processed it and turn those on instead, for example if you're processing months or days in a slider. That's how you'd do it.

NOW, if you save this and run it, you'll get an initial load of the map everybody already has for Lab 6, but if you use the slider, it will begin highlighting *only* those buildings selected in a given year. You'd need to write one more function to fix this (essentially an "initial date" function that tells it what to do if nobody's interacted with the timeslider yet). I was going to have you do that, too, but I wanted to make these optional additional labs more open. SO, for 20 extra credit points, change the example you've built here, but narrow the years to 1945-1960. If you read carefully through the instructions, you should have everything you need. Good luck!