**D3 Lab 2**

In today’s lab, we will spend more time with scaling and interactivity. Work through as much as you can in class, and finish on your own time. Submit by Wednesday Jan 19 at 10pm.

What to post to Canvas: Zip your HTML files and name the zipped folder with your name. Name each individual HTML file with which part it corresponds to (part1.html and part2.html)

Thanks to Jessica Hullman for some portions of this lab.

**Part 1**

D3 scales make certain mapping operations much easier. Remember that much of what we do in visualization is mapping from some input variable (the domain) to some output (the range). The domain can be *nominal*, *quantitative*, or *ordinal*. Similarly, the range can also be *quantitative*, *nominal*, or *ordinal*.

Some examples:

* We have weights ranging from 100 to 300 (the domain is [100, 300]), and we want to map it to an x coordinate ranging from 10 to 1000 (the range is [10, 1000])
* We have unemployment levels ranging from 0 to 10 and we want to map them to colors ranging from white to light blue
* We have temperatures ranging from -32 to 32 and we want negative temps to be mapped to blue, 0 temp to white, and positive temps to red
* We have nodes with a power-law degree distribution ranging from 0 to 1,000,000 and we want to rescale those to a range of 0 to 15 based on the square root of the degree

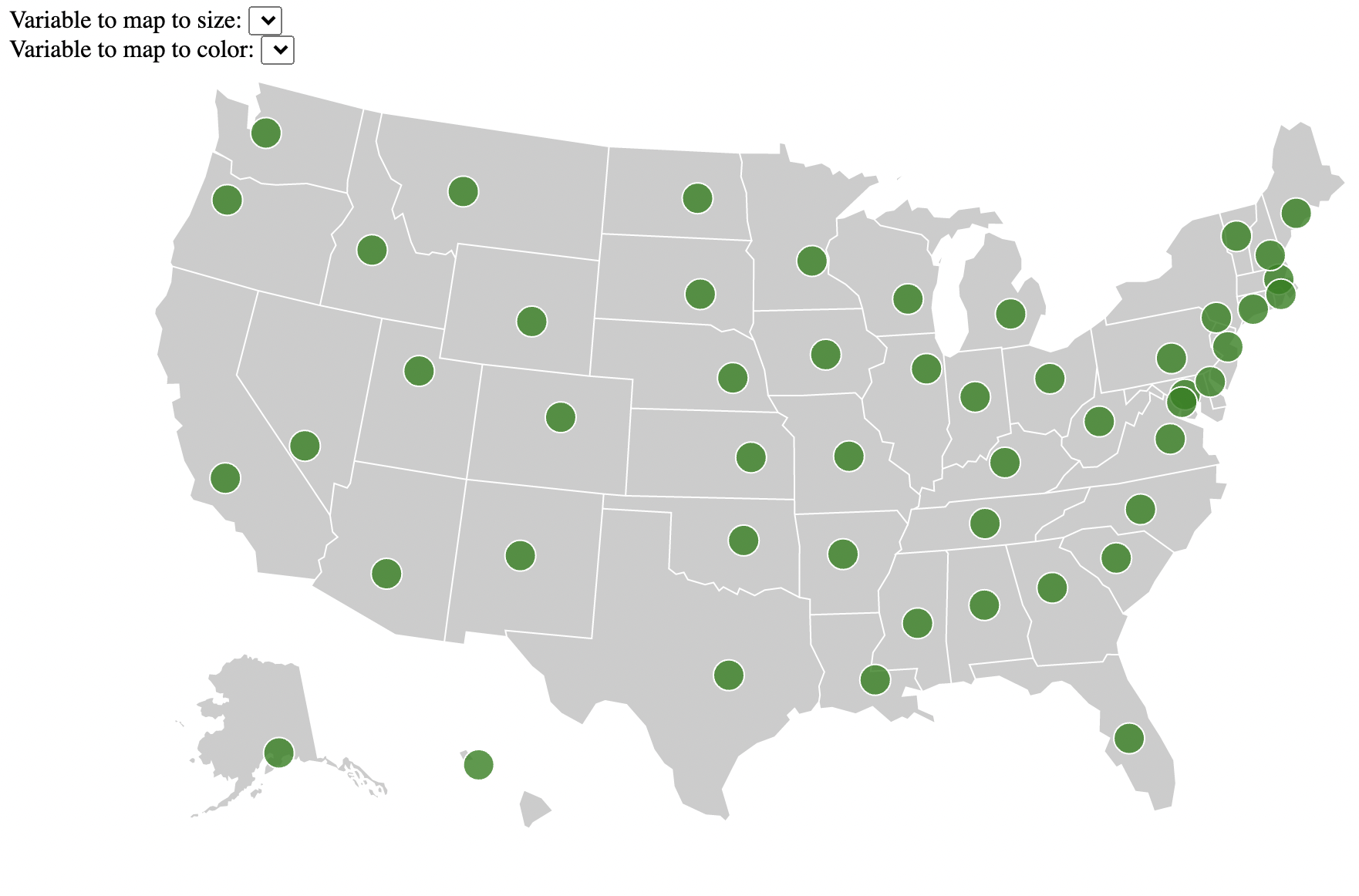
Load the template file part1.html into your editor. There are 4 rescaling questions. Implement the code for these like the example at the top of the file. Do not change the variable names. To test your code, load up the javascript console and run the sample commands to make sure you did the right thing.

[Here](https://github.com/d3/d3/blob/master/API.md#scales-d3-scale) is a link to the documentation for the D3 scaling functions.

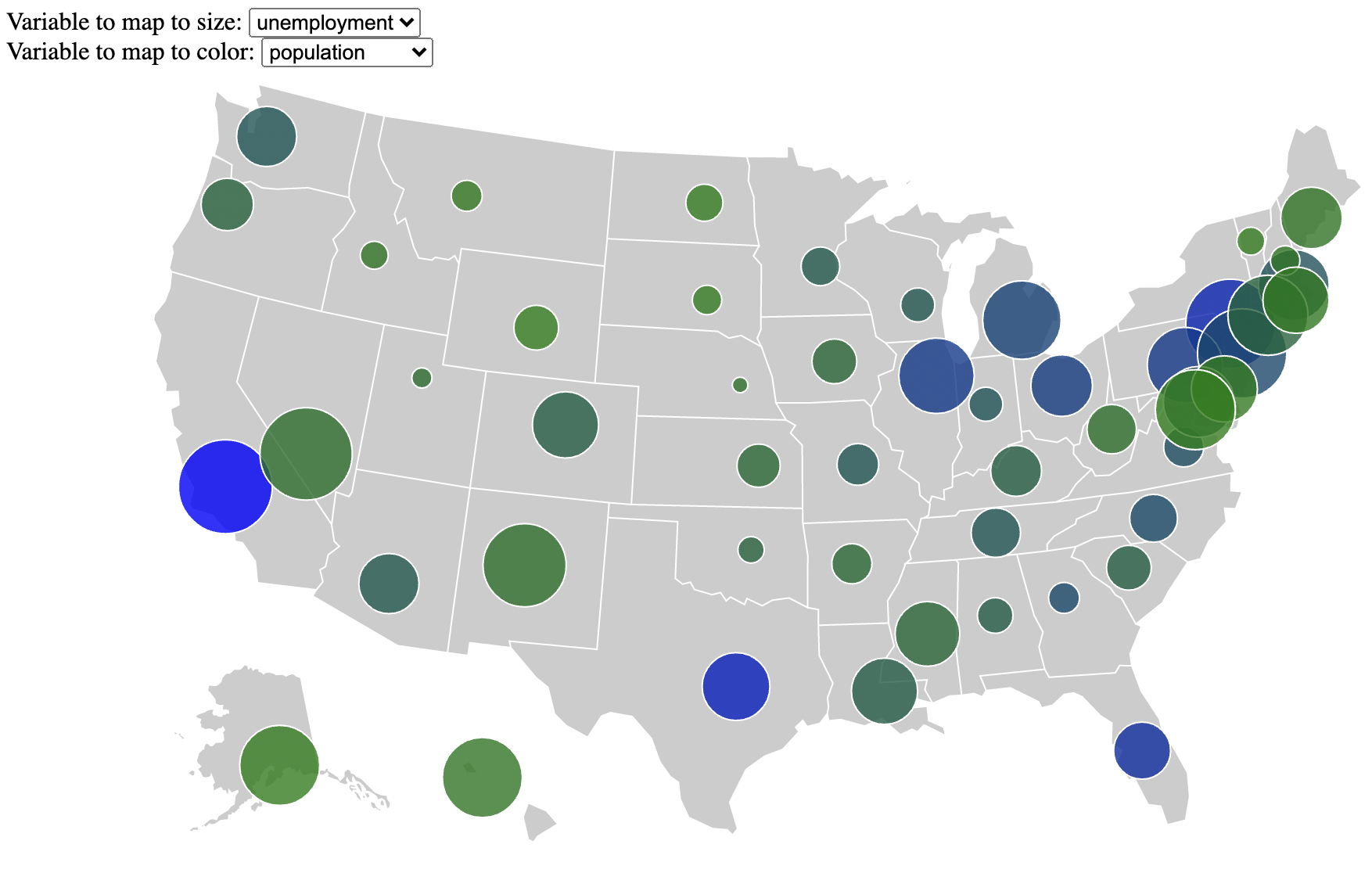
**Part 2**

Our goal in this part is to create a map of the US that visualizes the population and unemployment rate of each state. Above the map there are drop-down menus, where the user can select which of these two variables to display, and whether to map them to the size or color of the circles on each state.

1. Follow the instructions from Lab 1 to create a local server, and get the code running. You should see the following in your browser.



1. Add code for the dropdown so that users can select “none”, “population”, or “unemployment” for each of them. Whichever option they select should update the corresponding global variable sizeVar or colorVar. [This link](https://www.d3-graph-gallery.com/graph/line_select.html) has a helpful example of how dropdowns work in D3. Use the console to make sure your global variables match the values in the dropdowns.
2. Modify the drawCircles function to update the size and shape of the existing circles based on which options are selected in the dropdown. In order to do this, you’ll need to use the scaling functions you practiced working with in Part 1, and it will likely make sense to have separate functions determining what the size and shape should be based on the dropdown values. Some hints:
   * When mapping values onto circle sizes, it is better to scale circles according to their area than their radius. A circle's area is proportional to the square root of its radius.
   * You'll want to modify your dropdown code to also call drawCircles whenever their values are changed.
   * d3 scale functions have a way to deal with "unknown" values. You do not need to use this for your solution, but using it can make your solution simpler.
3. Here’s what your graph should look like with unemployment mapped to circle size and population mapped to circle color



1. Add a transition so that when you change either dropdown, the change is animated over a duration of 500 milliseconds. Recall how we did transitions in the previous lab.
2. Display the name of the state, population size, and unemployment rate below the map for any state you mouse over (look up the documentation for .on(‘mouseover’)).
   * Be warned: make sure you find examples for the latest version of d3 (version 6 or 7). Older versions handle mouse events slightly differently. You might want to look at the section on "event handling" in this document: https://observablehq.com/@d3/d3v6-migration-guide
3. Submit a zip file containing part1.html and part2.html