CS/INFO 3300; INFO 5100
Homework 7
Due 11:59pm Tuesday, October 29

Goals: Practice using d3 to create a choropleth map

Your work should be in the form of an HTML file called index.html or index.htm with one <p> element per problem. For this homework we will be using d3.js. **In the <head> section of your file, please import d3 using this tag: <script src="https://d3js.org/d3.v7.min.js"></script> and import topojson using this tag: <script src="https://d3js.org/topojson.v3.min.js"></script>**

Create a zip file containing your **HTML file and associated data files** (i.e. ny_rent.topo.json) and upload it to CMS before the deadline. Submissions that do not include data files may be penalized. Your submission will be graded using a Python web server run in a parent directory containing your zip file contents along with many other students' submissions.

There are several places in this assignment that may be a bit tricky. Please reserve additional time to complete this one, even though it only has one official problem.

1. In this problem we will make a **choropleth map of median monthly gross rent for ZIP codes** in New York state. We have provided a topoJSON file for you to use to complete this assignment. This shapefile also contains **bonus properties containing the median monthly rent of each county**. To obtain these data, we made use of data tools provided by Stanford to find 2022 rent data from the US Census Bureau. Using Python, we integrated these data into a shapefile created by OpenDataDE (which they had processed from other census records). **Please note that several ZIP codes do not have data**, leaving some gaps in the eventual map we will make. These ZIP codes are not included in the topoJSON file to make things easier.

As with any other TopoJSON file, there is a wealth of data available. You will find an **outline of the state** at `dataset.objects.state` and the **zip code data** at `dataset.objects.zip_codes`. Bonus data have been integrated into the **properties attached to each zip code geometry entry**. For this assignment, you will be **coloring zip codes based on the `median_monthly_rent` property**. We will create a **sequential color scale** and **bin income levels into quintiles** so that they are easier to spot.

(quartiles = 4 bins :: quintiles = 5 bins)

A. In the HTML portion of your submission, create the following SVG canvas:

```
<svg id="choropleth" height="770" width="990" style="margin:20px"></svg>
```

The width and height have been computed so that the NY map fits nicely into the canvas. In a <script> tag, use **`await`** to load the **ny_rent.topo.json** dataset into a variable called "`nyrent`".

Inside of your async function, please:

- Create a variable, "`state`", that contains a **topojson.feature** for the `nyrent.objects.state` GeometryCollection.
- Create a second variable, "`zips`", that contains the **topojson.feature** for the `nyrent.objects.zip_codes` GeometryCollection.
- Create a third variable, "`zipsMesh`", that contains a **topojson.mesh** for the `nyrent.objects.zip_codes` GeometryCollection.
- Finally, create a `d3.geoMercator` projection fit to the size of your canvas, and make a `d3.geoPath()` **path generator** that uses this projection.

B. Now, build **a quintile sequential color scale** for the `median_monthly_rent` variable:

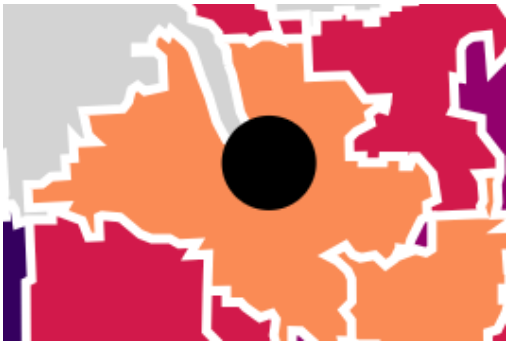
- First, **pick out 5 colors** for your sequential scale. Create an **array that has 5 elements**, where each element is a color string (e.g. **`"#ef21ac"`**). Your sequential color scale should follow **best principles** for designing color scales (hint: consider varying both hue and luminosity). **Please do not recreate the color scale in the example image**. You are welcome to find inspiration for colors from scales you find online, though you must manually specify each color and cite them.

- Next, to figure out the domain for your quintile scale, you need to **obtain an array of all values in the dataset**. You will have to gather these data manually. All of the values are stored within elements of the `zips.objects.zip_codes.geometries` array. *For each* of those elements, the income is stored in `.properties.median_monthly_rent`. We recommend that you use `d3.map()` to easily loop through the values and compose an array that contains what you need. You can also use a traditional for or forEach loop to accomplish this goal if you'd like. Make sure to `console.log()` your output to check that it contains the expected elements.
- As you've now defined both the domain and range of the scale, create `d3.scaleQuantile()` scale which you will use to color each zip code region.

Construct your visualization. You should, in order:

- Create a `<g>` tag in your SVG element to contain map elements.
- As we have missing ZIP codes, we will use the state outline to make a grey region to fill in those gaps. Use a data join to create `<path>` elements for the items in `state.features`. Place it in your SVG canvas. It should sit **on beneath all your other elements**. Give it a **light grey fill color** so that the gaps caused by missing data will stand out.
- Next, use a data join to create `<path>` elements for each zip code in `zips.features`. Place these paths in your `<g>` tag. Use your quantile scale to set the fill of each path. You can find the `median_monthly_rent` value within the **bonus `properties` dictionary** for each feature. If in doubt, use the trick of successive `console.log()` statements to figure out what data you get in your `.attr` call, as demonstrated in class. Do not give these zip code paths a stroke.
- Use `.append().datum()` to create a `<path>` element for `zipsMesh`. Place it in your `<g>` tag. Give it a **1px white stroke** so it sits on top of your regions and visually separates them.

D. **Add a black circle of radius 5 to mark the location of the Cornell belltower in Ithaca.** The belltower is located at **latitude 42.4476** and **longitude -76.4850**. (hint: you can use your **projection** to determine the x and y pixel locations for the circle, see **d3.geo documentation** and be careful of your latitude/longitude parameter order).



(next page)

E. After the SVG canvas, **create a `<ul>`** (unordered list) element and **populate it with the thresholds for your quantile scale**, so that users can get an idea of how each color maps to numeric rents.

- First, check out the d3-scale documentation to learn how to **fetch an array of your scale's quantile thresholds**. Then, make an empty **`<ul>`** element in your HTML and give it an ID so you can select it in your code with D3.
- Finally, using **either a data join or a forEach loop**, iterate through the threshold numbers and **add one `<li>` (list item) tag for each threshold** showing its corresponding income level to the unordered list. Done correctly, you should have a **bulleted list with four numbers on it** (corresponding to the separation points between quintiles in your scale). You are welcome to customize and run **`d3.format()`** to style the numbers so they are prettier, but it is not required.

Part F of this assignment is completely optional.
Students who successfully complete part F will receive one free quiz grade drop. If you've missed a quiz, this is an easy chance to win back one of those 0 grades.

F. Add a mouseover interaction to your chart. It should include the following elements:

- When the user mouses over a filled zip code region, a small, filled box should appear near the shape (but not overlapping it). Inside of this box, list the ZIP code and the median rent. Use d3.format() to properly format the median rent so that it resembles "$1,270". It's okay if you pick a visual design like that of the October 9th notes.
- As the user mouses over a filled zip code region, adjust the additional mesh so that it creates an outline on top of the selected region (again, follow the October 9 bonus notes). To prove that you are using a separate mesh and not just cheating by setting the stroke-width of a zip-code path, please set the stroke-width to be 10px and the stroke color to black. This is obviously ludicrously large, but it will help to illustrate that you have done it correctly.