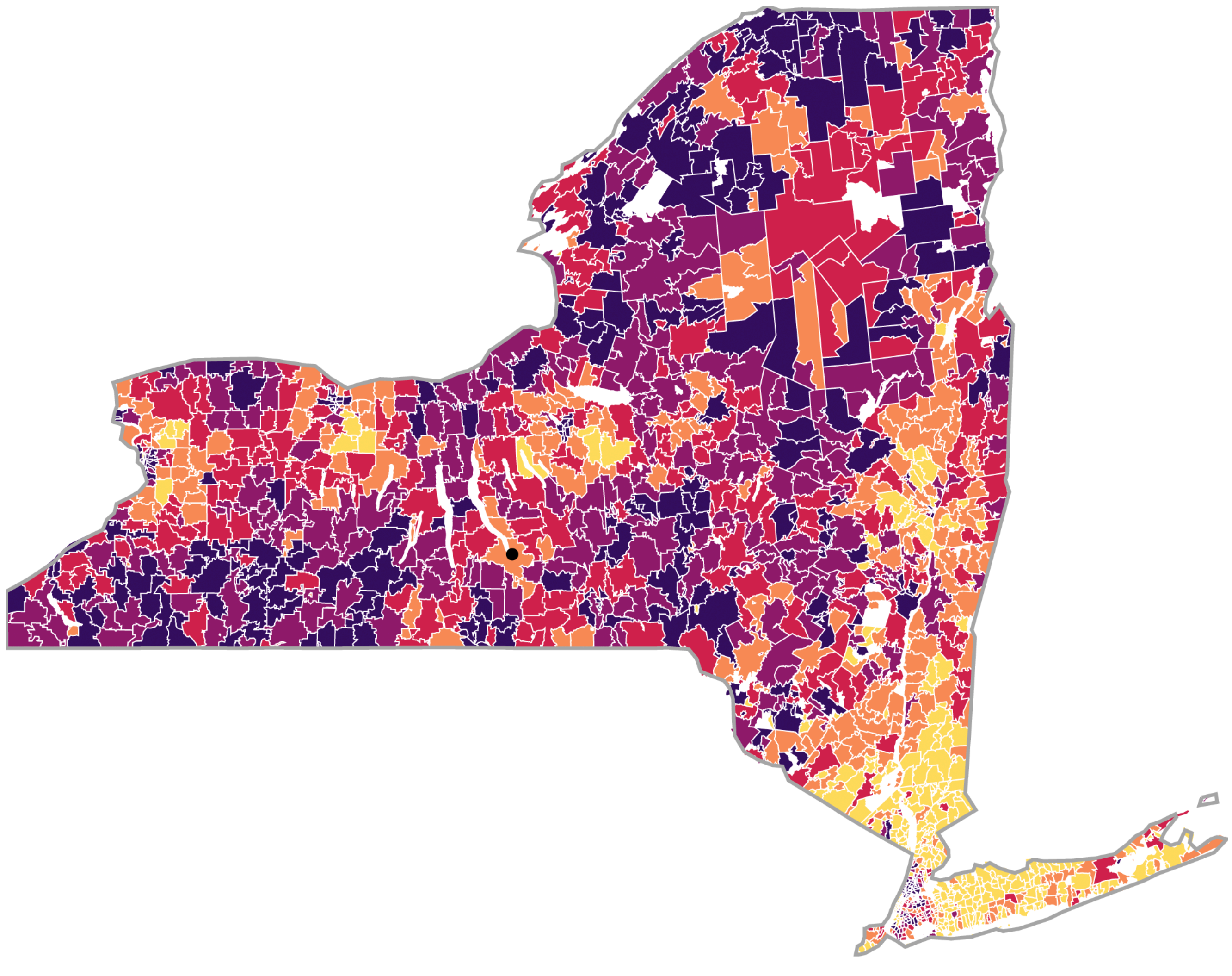
CS/INFO 3300; INFO 5100
Homework 7
Due 11:59pm Wednesday, October 26

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_income.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. Steps E and F are especially tricky.



1. In this problem we will make a **pan-zoom choropleth map of median family income 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 income of each county**. To obtain these data, we made use of US Census Data Tools to find 2019 median income data from the 5-year American Community Survey (pre-pandemic). Using Python, we integrated these data into a shapefile created by OpenDataDE (which they had processed from other census records).

As with any other TopoJSON file, there is a wealth of data available that can be quite confusing. 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** (e.g. `dataset.objects.zip_codes.geometries[0].properties`). For this assignment, you will be **coloring zip codes based on the `median_income` 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" />
```

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_income.topo.json** dataset into a variable called "`nyincome`".

Inside of your async function, please:

- Create a variable, "`zips`", that contains the **topojson.feature** for the `nyincome.objects.zip_codes` GeometryCollection.
- Create a second variable, "`zipsMesh`", that contains a **topojson.mesh** for the `nyincome.objects.zip_codes` GeometryCollection.
- Create third variable, "`stateMesh`", that contains a **topojson.mesh** for the `nyincome.objects.state` 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_income` 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 source color ideas from other 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. While this is conceptually similar to

class lectures, the Object.values() approach demonstrated in class is not applicable here. All of the values are stored within elements of the `dataset.objects.zip_codes.geometries` array. *For each* of those elements, the income is stored in `element.properties.median_income`. We recommend that you use d3.map() to easily loop through the values and compose an array. You can also use a traditional for or forEach loop to accomplish this goal if you'd like.

- 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 called `viewport` to contain map elements.
- Use a data join to create `<path>` elements for each zip code in `zips.features`. Place these paths in `viewport`. Use your quantile scale to set the fill of each path. You can find the `median_income` value within the bonus `properties` dictionary for each feature. Do not give these county paths a stroke.
- Use `.append().datum()` to create a `<path>` element for `zipsMesh`. Place it in `viewport`. Give it a 1px white stroke so it sits on top of your counties and visually separates them. Make sure to assign this mesh to a variable so that you can later edit the stroke-width.
- Finally, use `.append().datum()` to create a `<path>` element for `stateMesh`. Place it in `viewport`. It should sit on top of all of your other elements. Give it a 3px dark grey stroke to make state edges stand out.

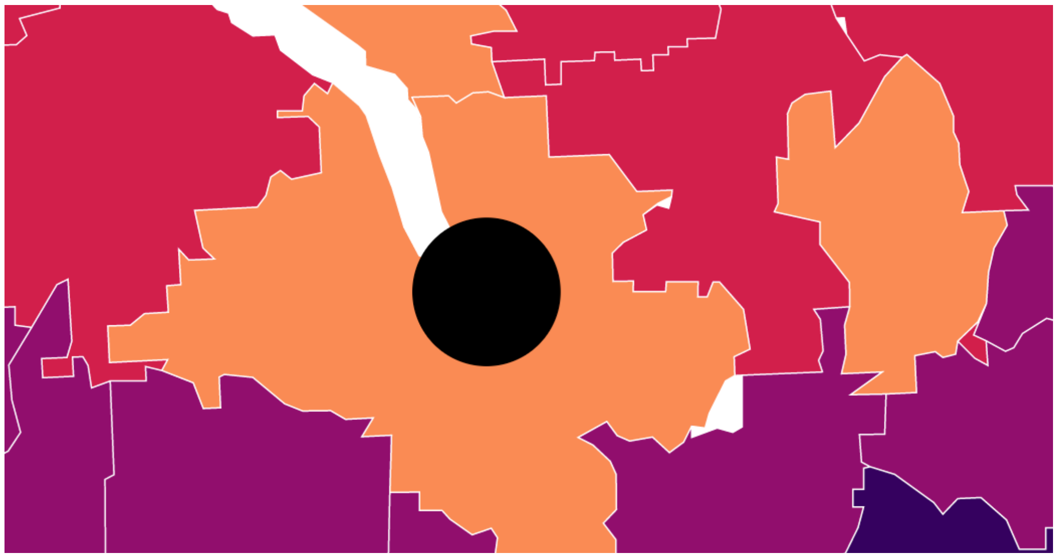
D. Add a black circle of radius 5 to mark the location of the Cornell belltower in Ithaca. Make sure it is inside `viewport.` The belltower is located at latitude 42.4476 and longitude -76.4850. (hint: you can use your projection to determine the x and y pixel locations for the circle, see d3.geo documentation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

E. Now that you have made the map, verify that it looks like the example image.
We are now going to make it have pan-zoom functionality. You should, in order:

- Make a `d3.zoom()` generator. Use `.scaleExtent()` to set its minimum zoom to 1 and maximum zoom to 10. Use `.on("zoom", func)` to bind a function to the zoom handler's events.
- Call your zoom generator on your whole SVG to make it zoomable.
- In your zoom event function, do two things. First, assign the transform provided by the event to your viewport so that items zoom in and out properly. Second, modify the `stroke-width` of your zip code mesh so that it remains 1px wide regardless of the zoom level (hint: divide by the zoom factor, `transform.k`).

(proceed to next page)

If you zoom in on Ithaca, you should get something like this if you have completed step E properly.



F. 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 incomes.

- 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.