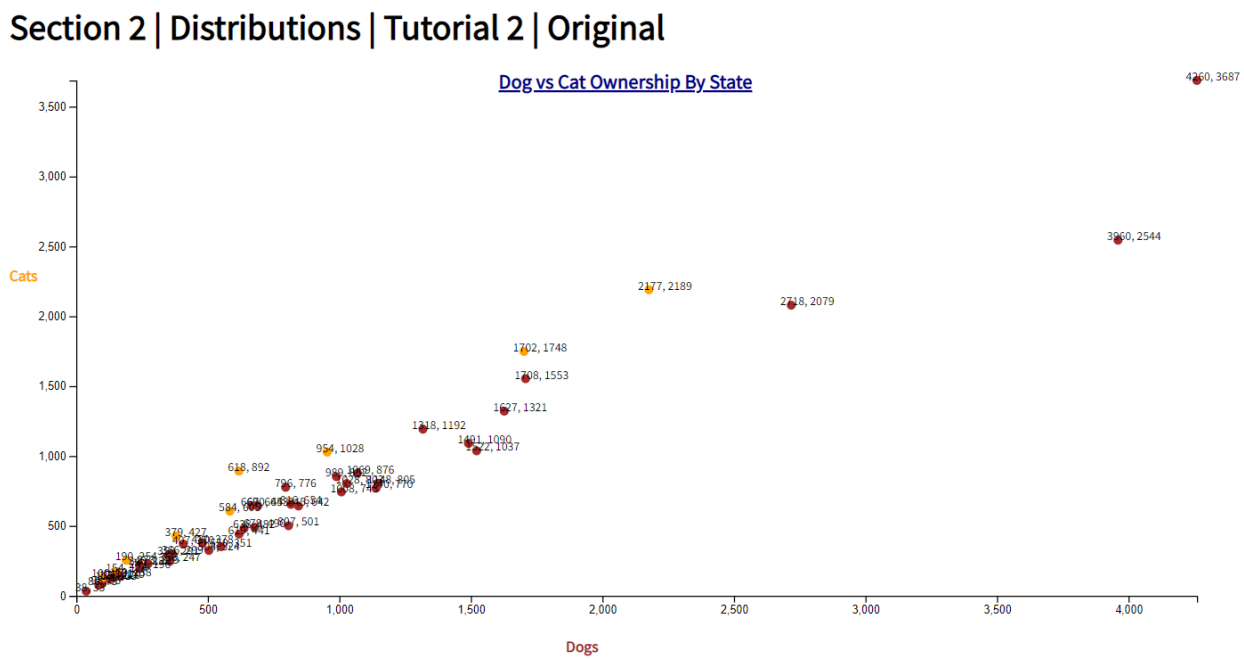
Nastasia Pollas

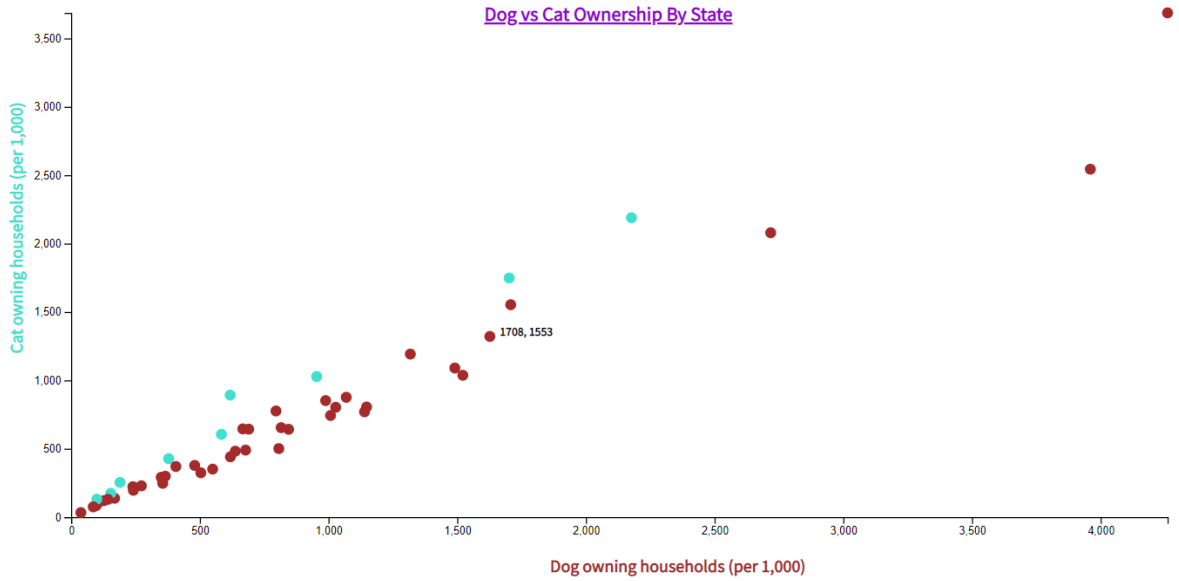
Updated section 2 tutorial 2 distributions scatterplot

Abstract
The goal was to learn how to plot a static scatterplot. I utilized a dataset called cats vs dogs, which I am honestly unsure where I got it from. Most likely it came from kaggle.com. The dataset contains statistics on cat and dog ownership for the contiguous United States. The columns include number in households per 1,000; percentage of households with pets; number of pet households in 1000; percentage of dog owners; dog owning households per 1000; mean number of dogs per household; dog population in 1000; percentage of cat owners; cat owning households; mean number of cats; and cat population.

Data manipulation
Joe and Katie helped me prepare the csv file for D3. First, I replaced the spaces in the column names with underscores and they helped me with the if statement that would help color the scatterplot. I compared dog owning households per 1000 vs cat owning households. I created a variable, called Dogs_or_Cats and ran an if statement - if value of D > C, then print D else print C. This would be used to color the distinguish the dots (brown for dogs and orange for cats). When creating the dot and joining the data, I used the Location header (states) as the unique ID.



Section 2 | Distributions | Tutorial 2 | Revised



Updated section 2 tutorial 3

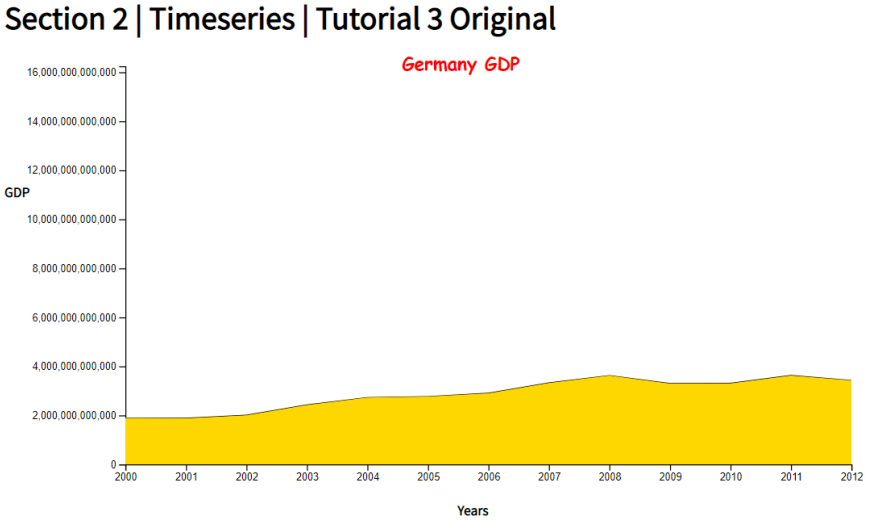
Abstract
The goal was to create an area chart, using a dataset of our choice, and to learn how to filter and group data, and to use the line generator function. Using one of Tableau's default dataset, World Indicators, I used the country, year, and GDP columns to assess changes in national GDP over a span of a dozen years.

Data manipulation
First, I parsed the data to convert the year from a string into a JS date. Then I converted the GDP column from a string into a number using +d.gdp. Finally, I returned the County header as a lowercase variable (country). Towards the end, I realized I had to sort the country column alphabetically on the original dataset so that it would appear sorted in the dropdown list.

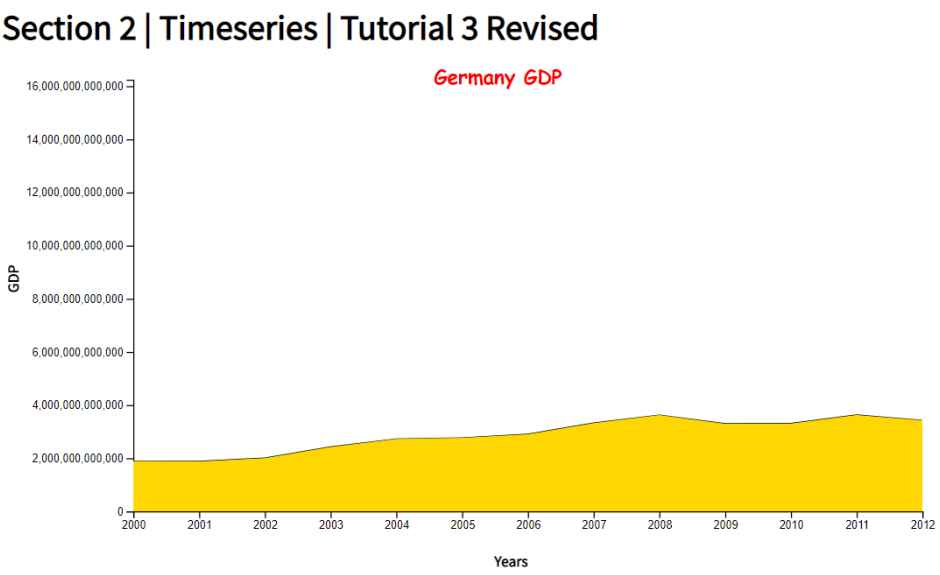
Img 1 – console log – this is how I applied the title to the static map

```
filtered
▼Array(13) i
  ▼0:
    country: "Germany"
    gdp: 1886400000000
    ▶year: Sat Jan 01 2000 00:00:00 GMT-0500 (Eastern Standard Time) {}
    ▶[[Prototype]]: Object
  ▶1: {year: Mon Jan 01 2001 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 1880890000000, country: 'Germa
  ▶2: {year: Tue Jan 01 2002 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 2006590000000, country: 'Germa
  ▶3: {year: Wed Jan 01 2003 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 2423810000000, country: 'Germa
  ▶4: {year: Thu Jan 01 2004 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 2726340000000, country: 'Germa
  ▶5: {year: Sat Jan 01 2005 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 2766250000000, country: 'Germa
  ▶6: {year: Sun Jan 01 2006 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 2902750000000, country: 'Germa
  ▶7: {year: Mon Jan 01 2007 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3323810000000, country: 'Germa
  ▶8: {year: Tue Jan 01 2008 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3623690000000, country: 'Germa
  ▶9: {year: Thu Jan 01 2009 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3298220000000, country: 'Germa
  ▶10: {year: Fri Jan 01 2010 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3304440000000, country: 'Germ
  ▶11: {year: Sat Jan 01 2011 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3628110000000, country: 'Germ
  ▶12: {year: Sun Jan 01 2012 00:00:00 GMT-0500 (Eastern Standard Time), gdp: 3425960000000, country: 'Germ
```

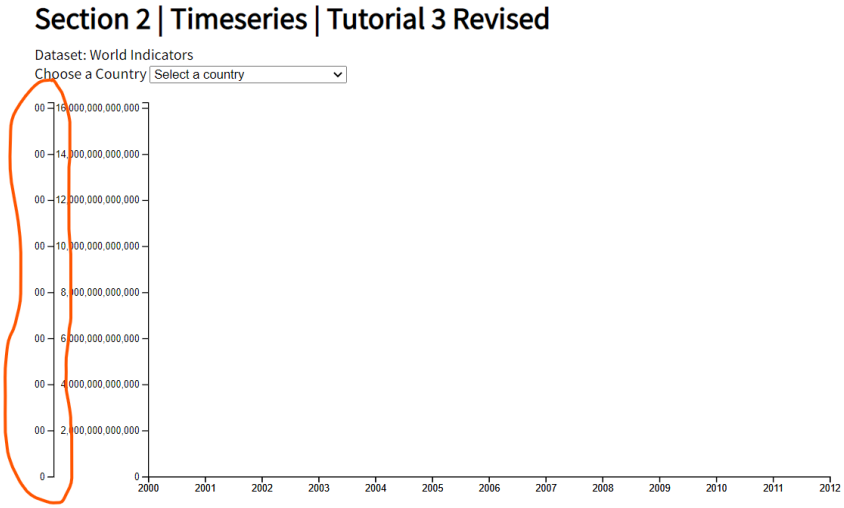
Img 2 - original



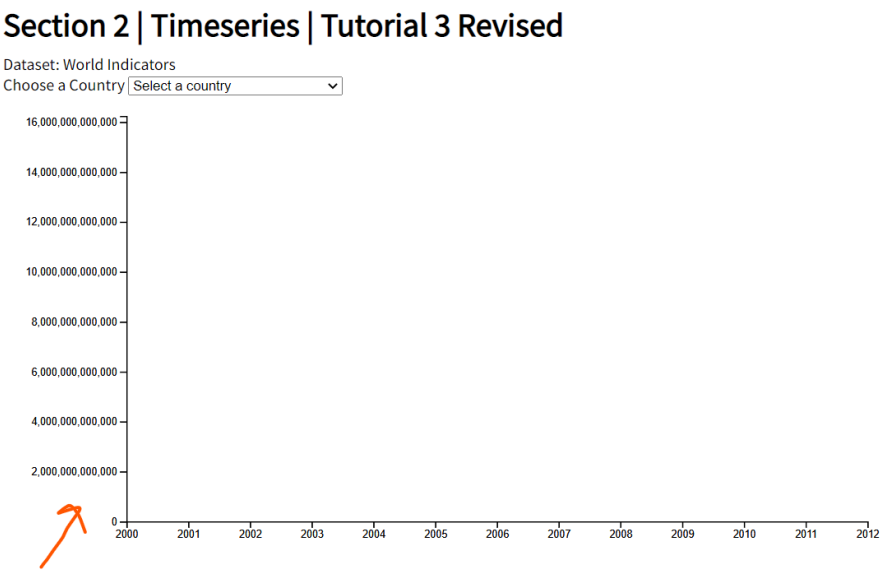
Img 3 – rotated y-axis



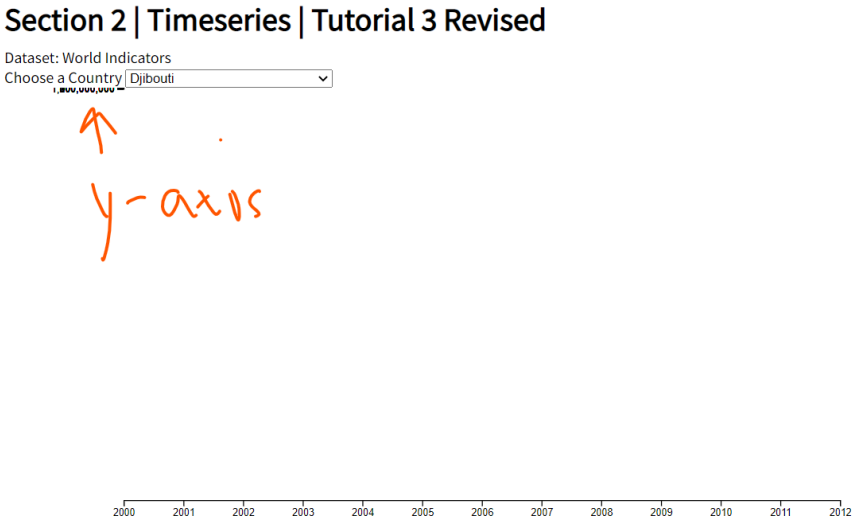
Img 4 – double axes



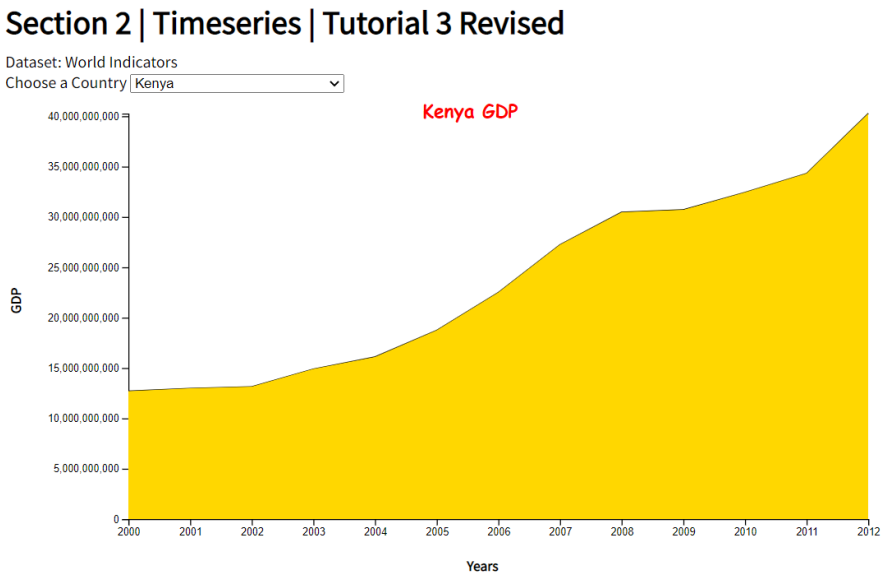
Img 5 – fixed y-axis (removed y-axis group)



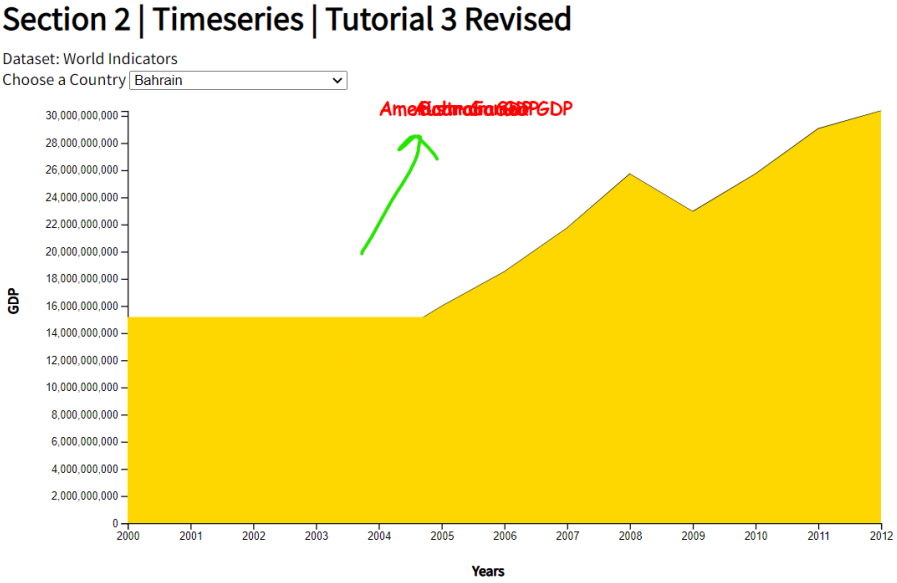
Img 6 – y-axis out of view



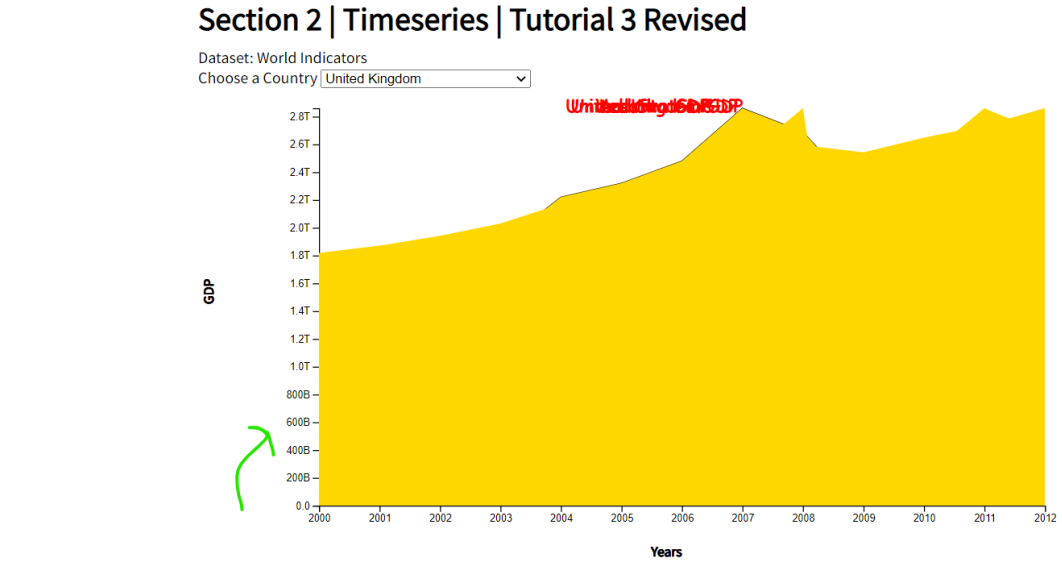
Img 7 – filter works and state updates



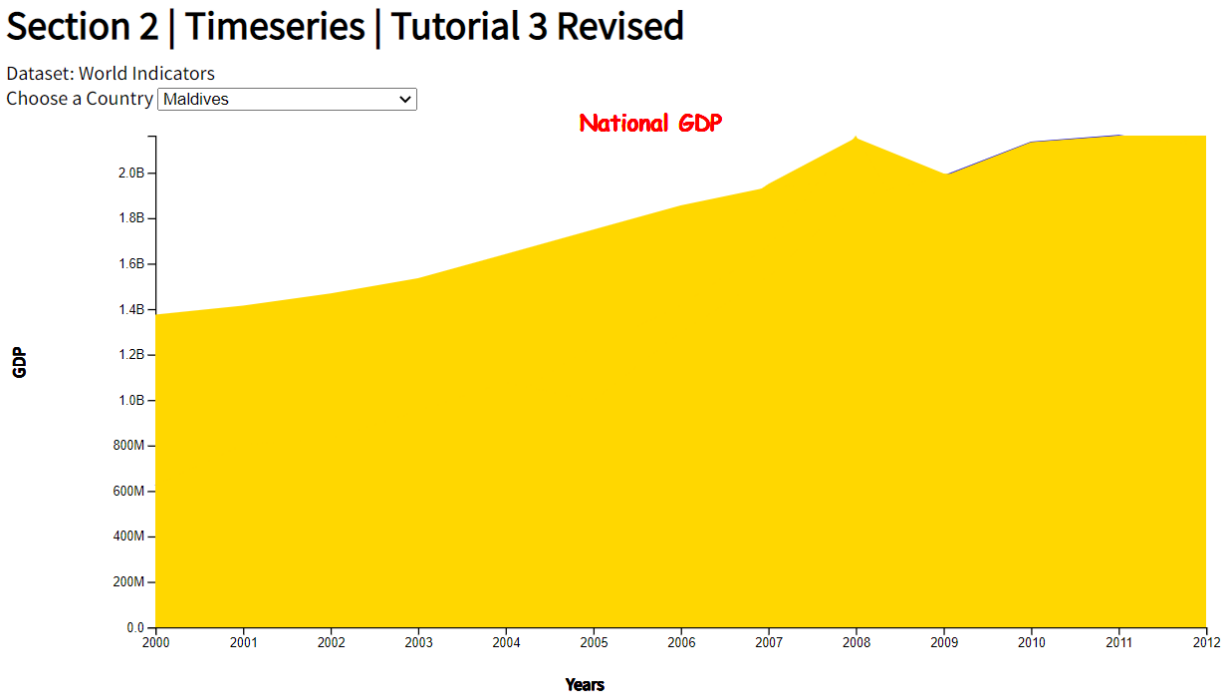
Img 8 – title overlaps



Img 9 – formatted numbers



Img 10 – final – generic title

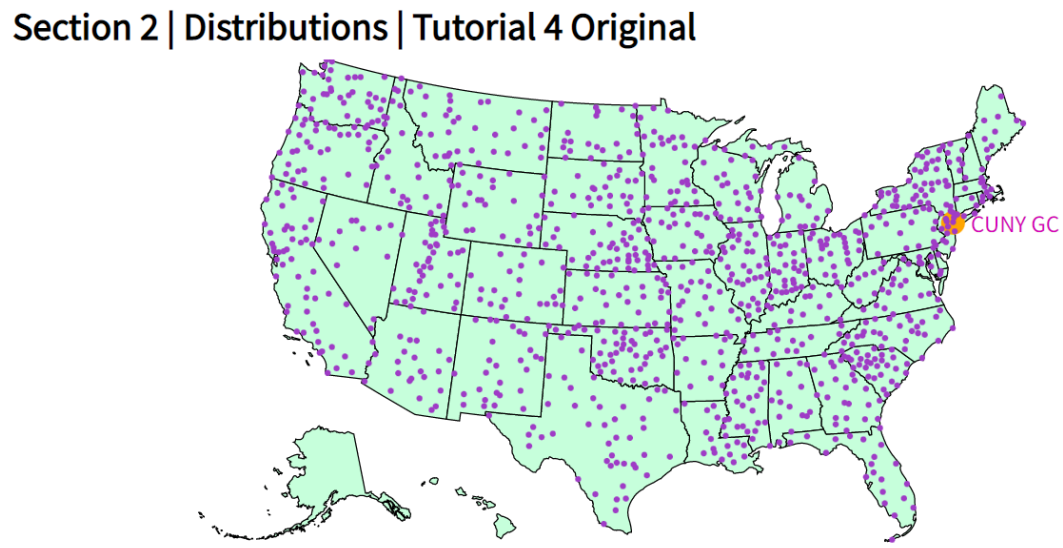


Updated section 2 tutorial 4 geography

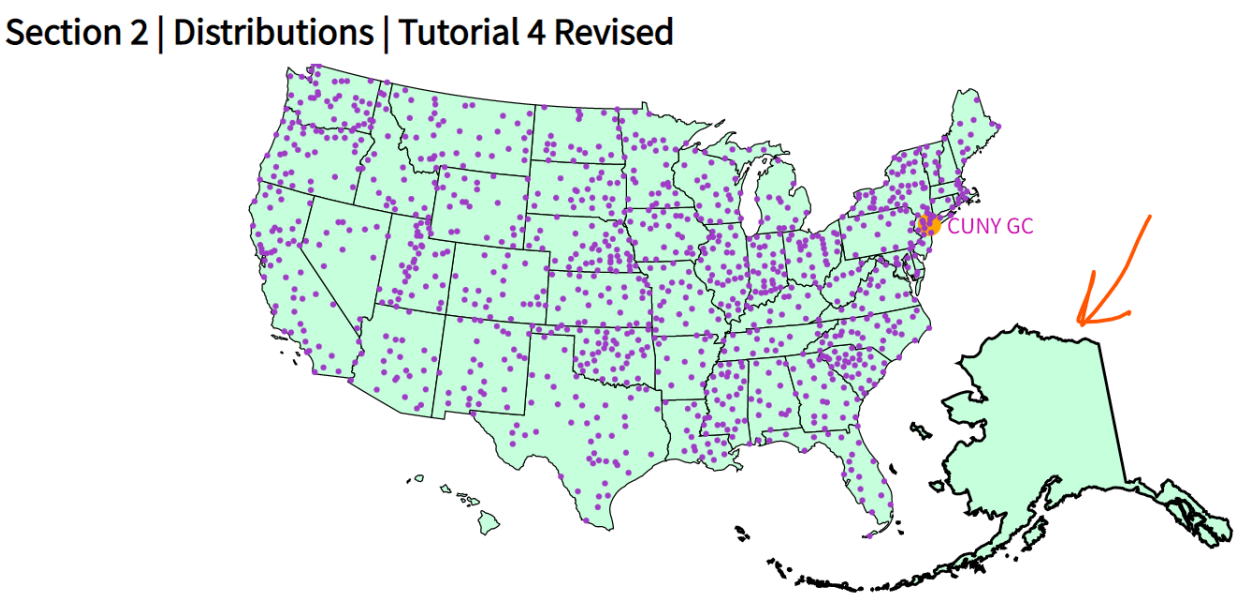
Abstract
The goal was to create a static map, plot heat extreme values as dots, and represent the location of CUNY Graduate Center with a distinct dot. We utilized the heat extremes.csv and the usState.json.

Data manipulation
The dataset consisted of mostly longitude and latitude coordinates, and I do not recall altering the data much or at all before uploading it with D3.

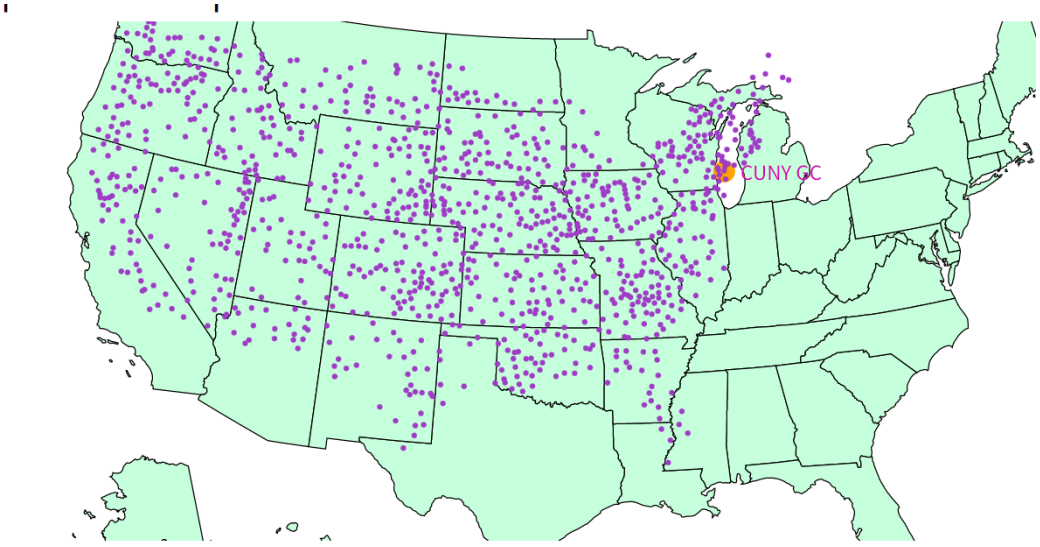
Img 1 - Original



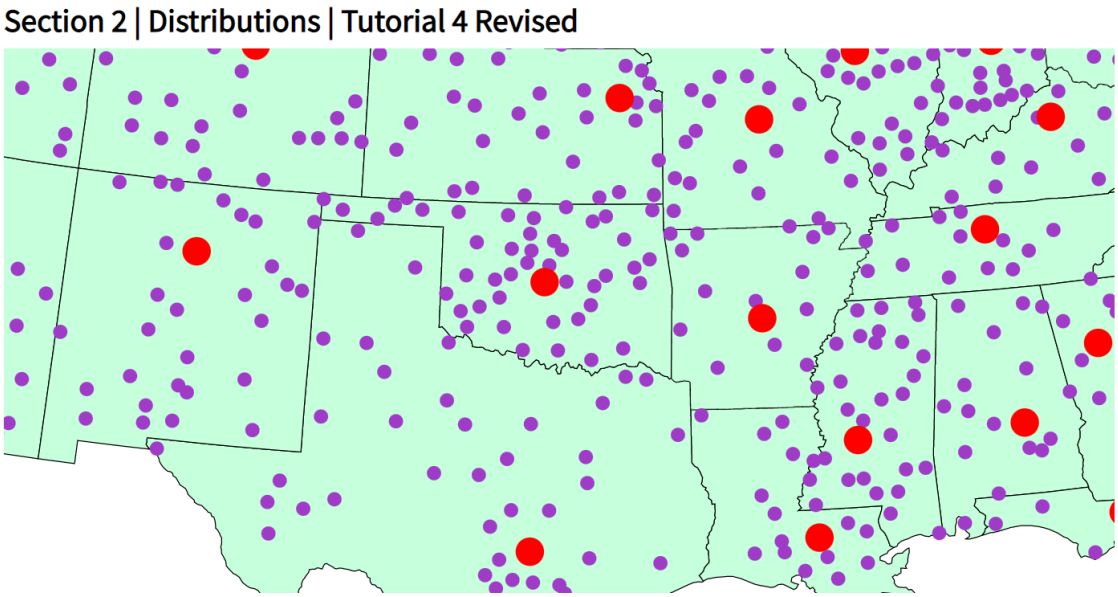
Img 2 - Alaska zooms and is movable.



Img 3 - States zoom but not the dots



Img 4 - Final - states and dots zoom and added capitals



Reflection

Section 2 distributions tutorial 2 scatterplot

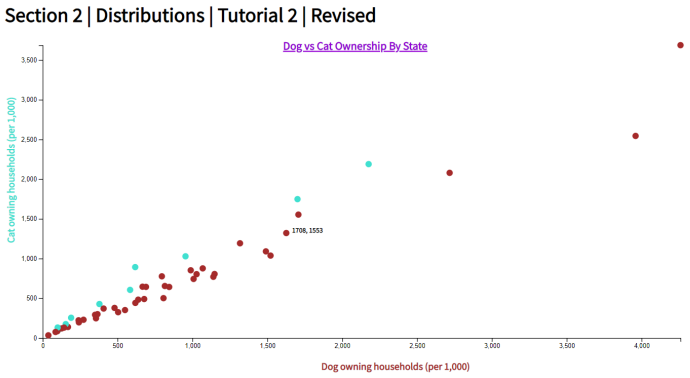
Original

In the original scatterplot, I knew little about how dot labels worked and tried unsuccessfully to move the labels above the dots. This resulted in a cluttered view, especially around the origin. I did not know how to rotate the y-axis label "Cats" either. Additionally, my axis labels of “Dogs” and “Cats” were not informative enough for viewers.

Revised

- Edited the axes label so they were more informative
    - Replaced Dogs and Cats with “Dog owning households per 1000” and “Cat owning households per 1000”
- Added mouseover function so when you mouseover a dot, the values – produces a less cluttered view
    - The dot labels appear below the dot instead of above them. I could not get the exact transformation correct.
- Rotated y-axis label
- Added transition using enter, update, and exit so the dots appear slowly on the screen
- Changed colors for y-axis label, y-axis dots, and chart title

- Changed the Cats dots to turquoise because the initial orange color blended too much with the brown at the origin



Section 2 distributions tutorial 3 time series

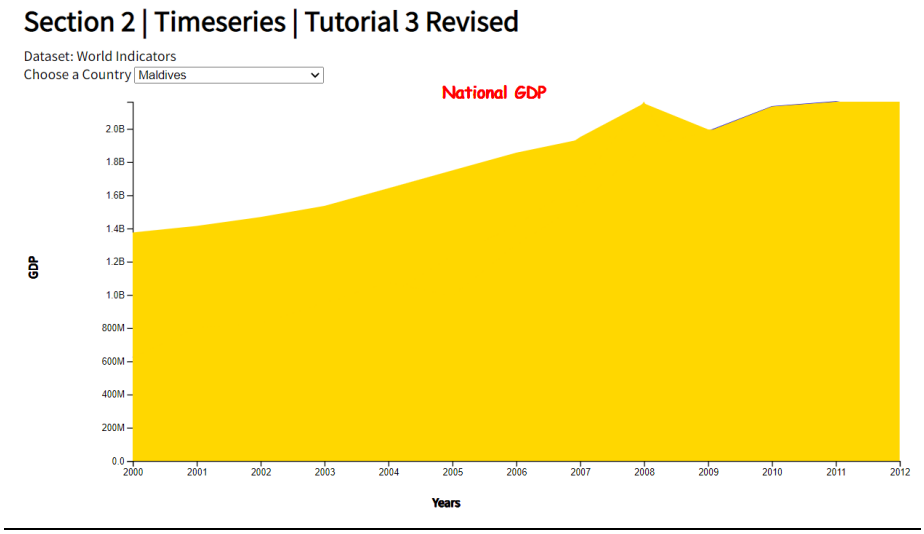
Original

The original static area chart would change if one updated the country name in this specific line: const filteredData = data.filter(d => d.country === "Germany"). The title would auto update to match the changed country because it was set to .text(`${filteredData[0].country} GDP`).

Revised

The goal is to create a filter where one can update the country on screen instead of editing the code.

- Rotated the y-axis label
- Repurposed the demo code from the 3.3 time series and the DOM seems to have 2 sets of yAxis on the screen.
- Commented out the 4 blocks of code containing the yAxisGroup & xAxisGroup, now DOM has the original x/y axes.
- Fixed the double axes issue by setting the same .attr("transform", `translate…) for yAxisGroup as the yAxis
- Then commented out the .append("g") and .attr for yAxis & xAxis
    - Results in a chart with fixed xAxis and yAxis that updates but is hidden away at the top left. The area chart still does not show.
- Had to rearrange the code to get the state working – moved the creating SVG down to right above the x/y Axis Groups section.
- Problems remaining
    - Countries not sorted alphabetically,
    - Countries' titles overlap at the top – title does not go away after next selection made
    - Area chart looks a little strange.
- Solution
    - Reuploaded dataset with alphabetically sorted countries,
    - Replaced with a generic title because trying to interpolate it so it would match the country from the filter did not work

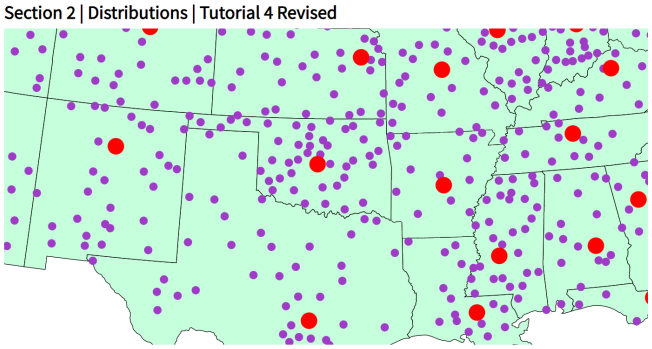


Section 2 distributions tutorial 4 geographic map

Original
The original is a static map with heat extreme points and a larger point to mark the location of the Graduate Center along with a label.

Revised
I wanted to make the map zoomable.

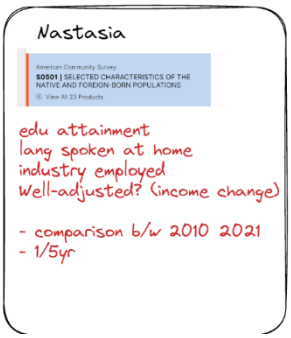
- Searched for appropriate code
- https://www.d3indepth.com/zoom-and-pan/ resulted in only Alaska being zoomable and movable across the map – removed this code
- https://observablehq.com/@d3/zoom-to-bounding-box?collection=@d3/d3-zoom
    - Appended "g" to the const states and the svg because the zoom codes seem to require "g"
        - Result – the states zoom but the dots are stationary
    - Replaced "svg.selectAll" for the .join("circle") and the gradCenterPoint with "g.selectAll"
        - Result – zoom works
- Also increased the SVG height so the map fills out more when zooming
- Added state capitals – map is a little cluttered, but I wanted to construct a layered map



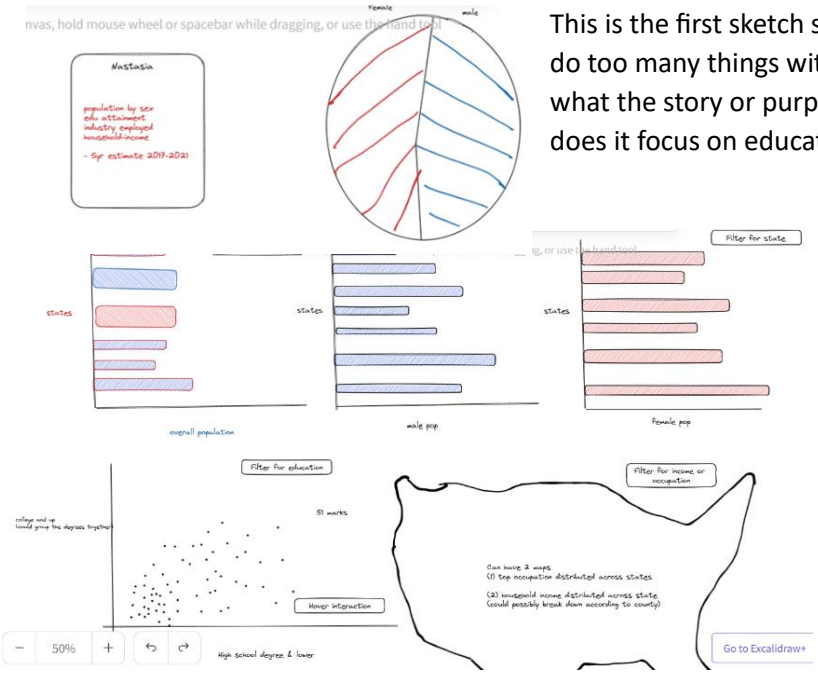
Census project

Critiques
The critiques were great – especially the suggestion to limit the range of the axes, using d3.extent, so the scatterplot would not cluster in the center. I also received a suggestion to be mindful of making things colorblind-friendly, so I used complementary colors (purple and yellow-orange). I also received a suggestion to try the stacked bar chart – I tried but I could not get the groups and data.columns.slice to work.

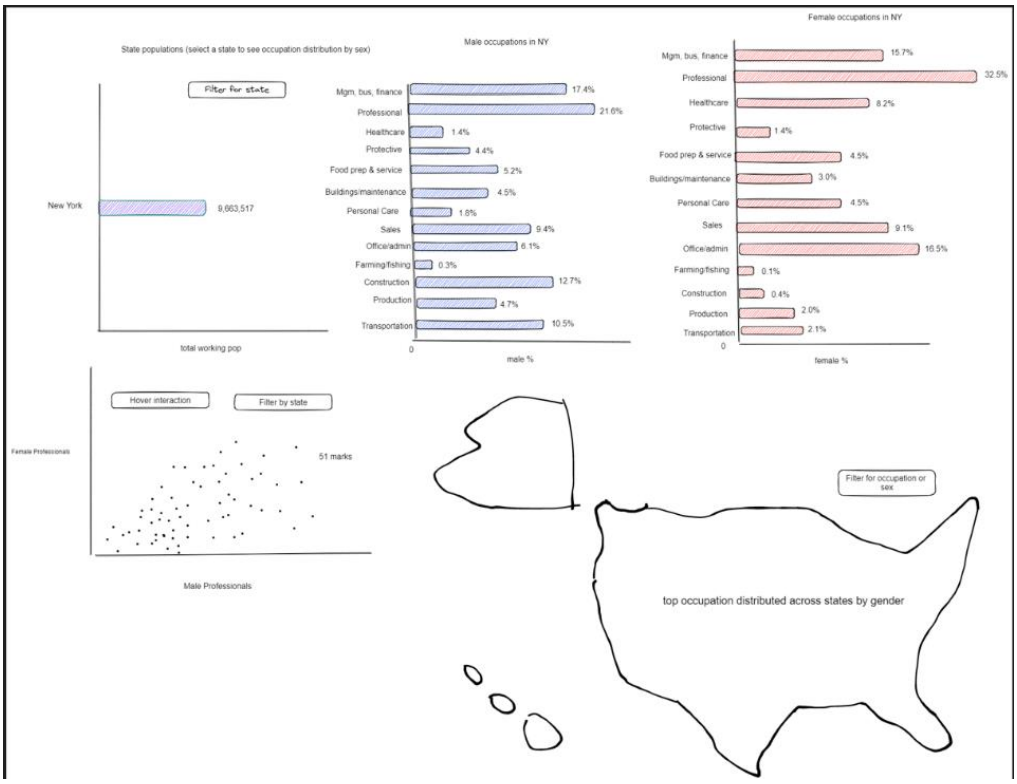
Differences between final and prospectus
The differences between where I started and ended are pretty drastic – in fact I created an entirely new prospectus and plan because the initial dataset had no real connection, in my opinion. I initially thought to look at data on income for native and foreign-born populations. (1st picture) Then, I decided to assess the financial picture of everyone, regardless of national origin, using education, median household income, and employment and poverty status. (2nd picture) The original dataset was too unwieldly, so I finally limited it to just the employment sector and occupation types of civilian men and women. (3rd picture)



Behold, the first ever ideation and draft on excalidraw!

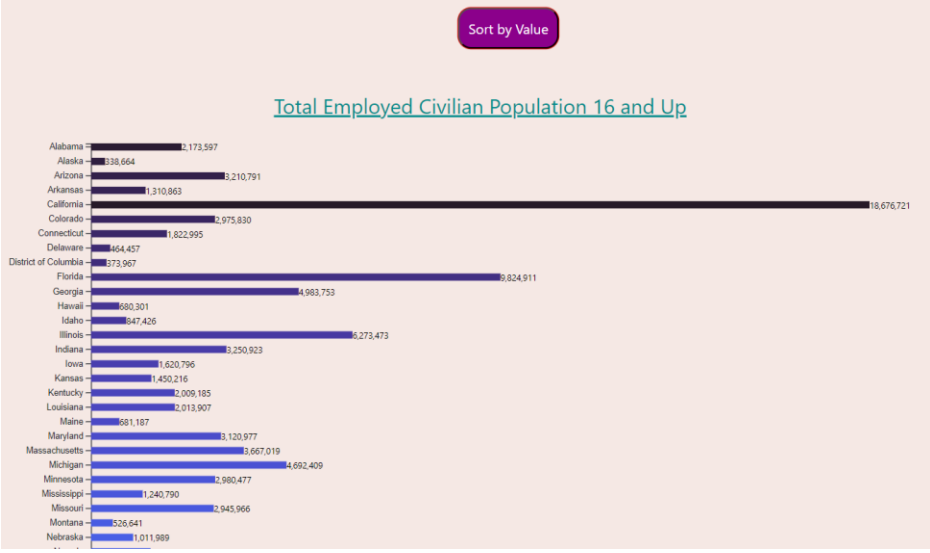


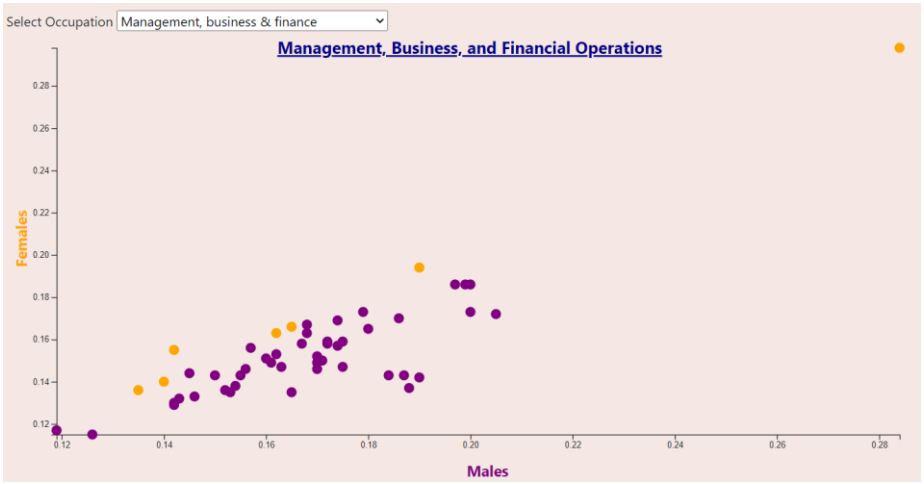
This is the first sketch sent; you can see that I was trying to do too many things with too many variables. I was unsure of what the story or purpose of this visualization would be – does it focus on education, income, occupation, or gender?



Final sketch – as one can see, I narrowed the scope of the data to just occupational categories per sex, per state.

The final output ended up being quite different than the final sketch. First, I created 3 bar charts showing the total population per state and then dividing that by sex. Then, I realized that this was irrelevant to the story. The total population includes those too young or old to work, and those who cannot or are not currently employed. So, I remade the bars and depicted the total employed civilian population per state and then divided that by sex. Unfortunately, I did not understand state management, nor did I realize that I should assign a different class to each bar. Only the first bar chart sorted, the y-axis remained stationary, and only the bar labels for the male and female charts. I created a scatterplot that showed the difference between males/females for the same occupation type, but I could not get the code right for state management and filtering. I eliminated the map, just could not do it.





Major challenges include:

1. Data cleaning – I probably cleaned and redid the dataset > 15 times because I was never sure if the states should be listed as the rows or as the columns.
    a. You can see that with the numerous versions of datasets in the data folder
2. Not understanding how the data is “shaped” – array of arrays, objects of arrays – what are those and why are they important?
3. Not understanding how the code worked – (ex: not understanding what inputs certain functions require)
    a. Still must understand how “passing through a function/argument” works
4. STATE MANAGEMENT!!!!! I could not get this – how do you apply state to multiple SVGs?
5. Could not get the filterData working for the scatterplot – console.log returned empty array
6. Could not group the different jobs according to sex for comparison

    a. This was the most frustrating because I thought that I could just take the columns from the dataset that divided each occupation type by gender, group them, feed them into the filter so they could compare.
7. Could not always format external code – such as from observable – which used things like .node or let chart = {}
    a. I found that a lot of observable notation was structured differently than class or demo codes, so it was hard to apply
8. Could not get the ascending code to work – bars could only sort in descending order

How to mitigate challenges in the future?

1. Learn state management – this would help so much
2. Understand how to interpret and apply observable or stacked flow code
3. Really understand developer tools and the console log – I had a hard time interpreting the console log errors and then not fully utilizing the element tab to locate elements when they would not appear
4. Fully learn GitHub – when it came to pasting the links, I was unsure of what to do exactly, so watching some tutorials would help
5. Assign a class to each element created

If I had more time and resources, I would

- Include a pie chart breaking down the different employment sectors for the civilian population (I tried, but I could not get the path correct.
- Resize the bar charts and set them next to each, like a grid, so it was easier to see the differences
- Get all the bars working with 1 button (I have updated the census project (in a different location) and the bars all sort but each needs its own button)
- Create a legend for the scatterplot (I did in a separate version afterwards) but I must learn how to move it because it overlaps on a mark
- Create a map of occupation distribution
- Filter out DC because it was an outlier for lots of occupation types

Most helpful thing I learned from all the work

I learned so much about myself in this course, and the prospect of continuing to build on D3 excites me. But there is 1 lesson that has left an indelible impression on me.

- NEVER FORGET THE **4TH JONAS BROTHER!!!!!!!! –** I apologize for overlooking you, Frankie Jonas.