PSET 7: Building a D3 Visualization

We've now covered the basics of D3, including how to bind data and draw shapes to create simple visualizations. Combined with web fundamentals (building a site, HTML, CSS) you should be able to build standalone D3 visualizations. The D3 material we've covered includes:

- Binding data to shapes
- Using `.data()`, `.enter()`, `.append()`, `.attr()` and more
- Using anonymous functions
- Declaring a dataset as a variable
- How to change styles on your visualization
- Different shapes you can use to create visualizations (circles, rectangles, lines, etc.)
- Changing the width and height of your SVG
- Adding text to your visualization

Exercise 7.1: Customizing a basic D3 visualization

First, open **d3psetindex.html** in your text editor and web browser. You will be modifying this file to customize a D3 visualization.

About the visualization
This visualization we're creating draws from World Bank data on agricultural land in a set of countries in 2016 (as a percentage of total land). The following is the data you'll see in the <script> tags of the d3psetindex.html file:

```
const agricultureData = [["Brazil", 33.9],["Canada", 6.9],["Costa Rica",
34.5],["Denmark", 62],["Fiji", 23.3],["France", 52.4],["Greenland", .6],["Italy",
43.2],["Mali",33.8],["Netherlands", 53.3]]
```

Notice that we're adding the D3 visualization to the div with ID="viz". This is defined in the following line of code:

```
const svg = d3.select("#viz")
```

The subsequent D3 code adds rectangles ("rect") to the SVG and sizes them according to the data (agricultureData).

In line 47, you will see that there is a group element appended to the svg with the

`.append("g")` method before the rectangle elements are populated with the data. Creating groups for similar elements is a way to help organize your code in the DOM structure. This will be necessary when we want to create multiple text elements that respond to different aspects of the data.

```
46          // add rectangle bars
47          svg.append("g")
48              .attr("class", "rect-bars")
49          .selectAll("rect")
50              .data(agriculturedata)
51              .enter()
52              .append("rect")
53                  .attr("y", (d, i) => i * 30 + 30)
54                  .attr("x", 50)
55                  .attr("width", d => d[1] * 2)
56                  .attr("height", 20)
57                  .attr("fill", "orange");
```

The current version of the site should look like the following:

You will be customizing the above chart to make it more clear and visually appealing. Complete the following tasks:

Task 7.1.1: There are predefined variables at the beginning of the JavaScript section of the document. Implement and update these variables in your D3 code in the following steps. For example, the width is currently defined as `d => d[1] * 2` and can be updated to use the variable `widthMultiplier` instead of the value `2`.

Task 7.1.2: Change the color of the bars to green and their opacity to 0.7

Task 7.1.3: Triple the width of each bar, set the height of each bar to 30, and set the spacing between bars (gap) to 5 (you can also change the dimensions of the SVG as needed).

Task 7.1.4: Use the margin variables to move the bar chart to the right by 100 pixels and downward by 50 pixels.

Task 7.1.5: Add a group and text labels to the bars specifying country names.

- Use a CSS class for the text styling and name the class “text-styling” - see the Google Fonts reference page for help on styling.
- Use Open Sans font, weight 300, and size 14px for the text styling
- Be sure to right-align the labels (*hint:* Use the “text-anchor” attribute in D3 to right-align the country names).
- You may need to move your whole bar chart to the right to allow room for the labels.

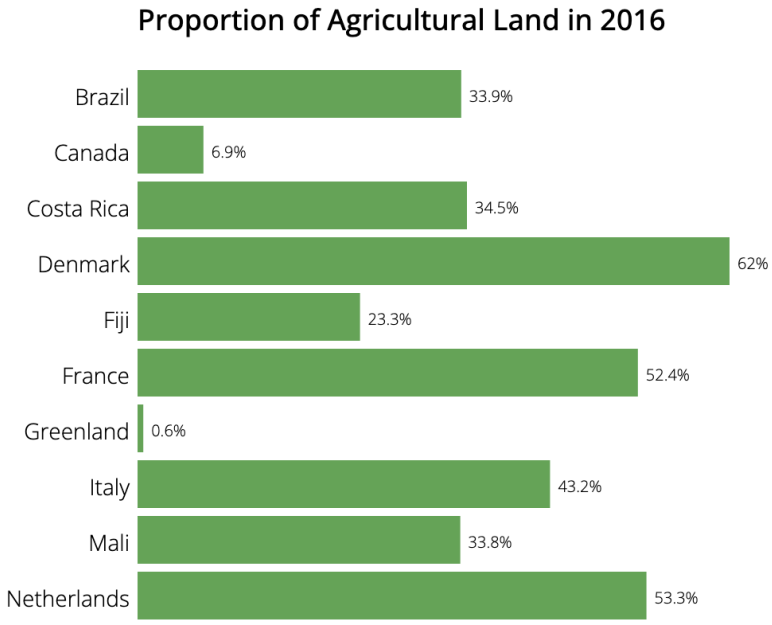
Task 7.1.6: Add a group and text labels to the bars specifying the percentage of total land per country.

- Use a CSS class for the text styling and name the class “label-styling”.
- Use Open Sans font, weight 300, and size 14px for the label styling
- Be sure to right-align the labels (*hint:* Use the “text-anchor” attribute in D3 to right-align the country names).
- Be sure to add a `%` symbol to the label so the value is easier to understand

Task 7.1.7: Add a title to the chart.

- Use a CSS class for the text styling and name the class “title-styling”
- Use Open Sans font, weight 500, and size 18 for the title styling

Your final result should look similar to the following:



Exercise 7.2: Laying out your webpage

Now that you have plotted your chart, let's give the rest of the web page some structure with bootstrap components.

Task 7.2.1: Within the `<main>` div, create a `<section>` with the bootstrap class "container". This will make your page responsive to different widths, center the section element with a standard margin on the left and right.

Task 7.2.2: Within the `<section>` div, create two divs with the class "row". Recall how the bootstrap grid system works with rows and columns in Tutorial 8 or via the Bootsrap Documentation.

Task 7.2.3: Within the first row, add a `<h1>` header element with the title "Visualizing Agricultural Land".

Task 7.2.4: Within the second row, create two divs with the class "col".

- Set the first column to a grid width of 5. This will be located on the left side and contain descriptive text.
- Set the second column to a grid width of 7. This will be located on the right side and contain the chart visualization. The `<div id="viz"></div>` element should be

located within this div.

Task 7.2.5: Populate the left column with a paragraph of text, followed by our course information and your name. (Placeholder text is OK for this exercise; see example below for the text elements)

Task 7.2.6: Add a paragraph element after the visualization to reference the data source, the World Bank.

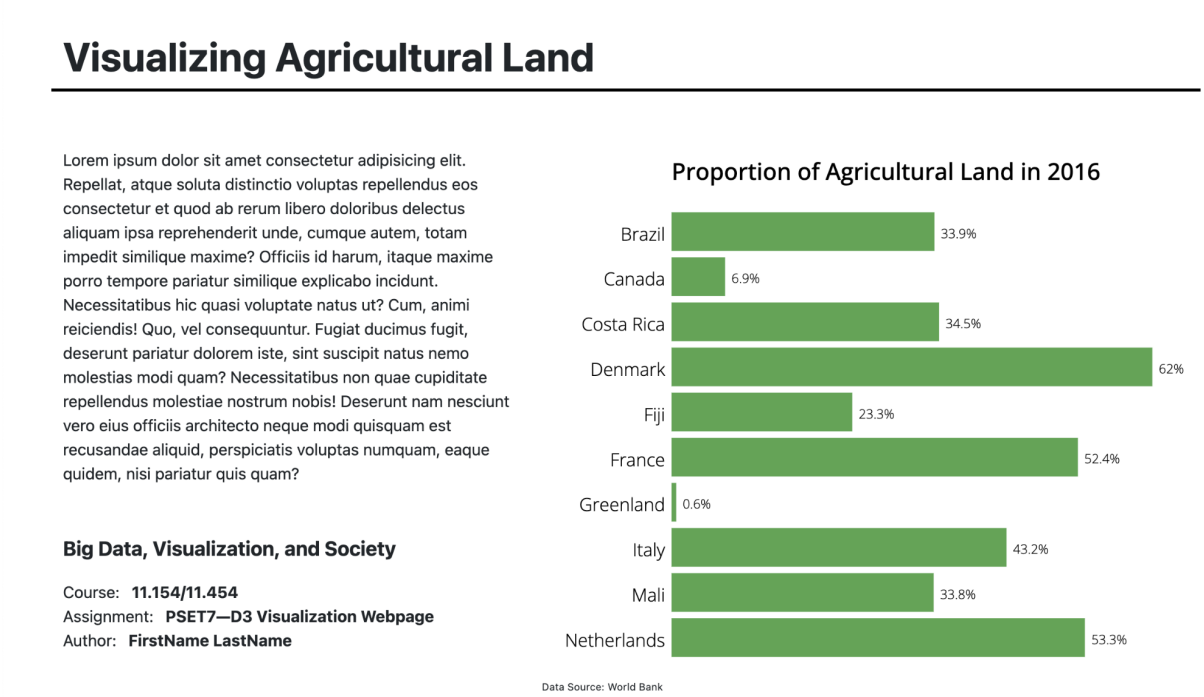
Task 7.2.7: Another way of defining the dimensions of your D3 svg is to use the "viewBox" attribute. Defining the height and width this way will help allow your svg to be responsive to different screen sizes (to make a truly responsive webpage further styling will be necessary).

- Redefine your svg with `.attr("viewBox", [0, 0, width, height])` instead of the width and height attributes.
- The array values for viewBox set the values for `x`, `y`, `width`, and `height` respectively in that order. `x` and `y` are set to 0 as the starting positions for the svg dimensions.

Task 7.2.8: Add style to your web page elements! You can use bootstraps built-in style utilities, such as spacing or text, or define your own CSS styles. Be sure to include the following minimum styling requirements:

- Style the title text in some way (font weight, size, capitalization, etc.)
- Add a border or background color to an element
- Create 3 levels of text hierarchy (implement at least 3 text styles that change font weight, size, capitalization, etc.)
- Add spacing between your elements so different elements on the page are distinct and clearly laid out

Your final result may look something similar to the following:



Deliverable: Please upload your file to your GitHub, host your website live (GitHub pages recommended), and submit a live url to your PSET7 visualization webpage.

Save above visualizations as an .html file, host the website live (via GitHub) and submit it on Canvas.