Tutorial 10: Interactive Visualizations
Leah Brooks, Data Visualization Using R
Summer 2022

Today we are going to turning away from R to study a new product that is better suited for interactive visualizations: d3. D3 stands for "data driven documents."

I am very grateful to many helpful webpages in building today's tutorial: D3.js tutorials, the Graph Gallery basic bar chart, and the Graph Gallery interactive bar chart.

1. D3 and Interactivity

We have already discussed the many virtues of a good static visualization. However, sometimes an interactive visualization is a better way to achieve visualization goals. Because of this, our final tutorial focuses on creating a few simple interactive graphics.

While R does have a tool for creating interactive graphics, it is somewhat clunky and the results do not look as nice as the D3 coding we'll learn today. In addition, interactive R graphics (built through R Shiny) are difficult to host on webpages, while D3 graphics are built directly for the web.

Because D3 graphics are built for the web, you write them inside a standard webpage. So we begin by writing a very simple webpage. For today's class, we are not going to post the pages you write to the web – though you could, if you have webserver access. But you can open them with your local web browser (Chrome, etc) and that is what we'll do to test them.

Everything we're learning today is good for data that are **already** in the form you want them to make a nice graph. So think of what we're doing today as a final step. You've done your data analysis and preparation in R, and now you want to display it interactively. D3 is spectacularly bad for doing any kind of data preparation.

In addition, there are many many little parts to a D3 graphic. We are not going to go over all of them today. I'm going to give your two overview examples and then ask you to modify one for the homework. You are very welcome to work on tweaking any components of the command, but you will not leave today's tutorial understanding every last element of the graph we'll create.

2. Get a HTML Editor

To make today's coding easier, I am recommending that you download and install Atom, which is a text editor designed for html. You don't need this per se – you could just write in Notepad or any other basic text editor, but Atom will make things in color, which as you know from this class, can aid in understanding.

You can find the download page for Atom here. If you have another editor you prefer, please feel free to use that instead.

3. Making a Simple HTML document

In this section we make a basic webpage.

3.1. Making a basic document

Open Atom, go to the file menu, and choose "New File."

Every html page begins and ends with

```
<!DOCTYPE html>
</html>
```

Note that this html command – and all html commands – are always in inside carrots (<>). Commands begin with some text inside carrots < `[some text]` >, and end with carrots with a forward slash <\>. The <`!DOCTYPE html`></`html`> command that we start with is one such example.

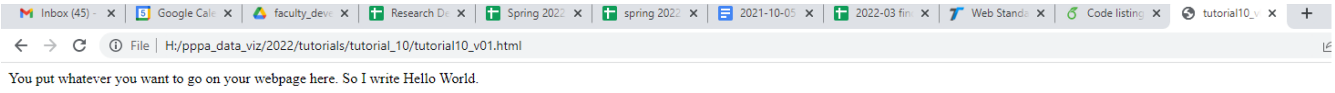
Any content goes between those two points. To make an exceptionally simple webpage, I can write a little bit of code in between that required beginning and ending.

```
<!DOCTYPE html>

You put whatever you want to go on your webpage here.  So I write Hello World.

</html>
```

Now having written this webpage, I save it with an html extension: .html. So, file → save `lecture10_v1.html`. I save mine in `H:/pppa_data_viz/2022/tutorials/tutorial_10/tutorial10_v01.html`. Make sure that your file has a `.html` at the end of the rest will not work properly!

Once that's done, open with your favorite local webrowser. To open in Chrome, I press <`ctrl`>+`O` and navigate to where I saved my file.

When my file opens, it looks like this:



H:/pppa_data_viz/2022/tutorials/tutorial_10/tutorial10_v01.html

You put whatever you want to go on your webpage here. So I write Hello World.

3.2. Adding a Shape

Before we add a graphic, we'll first add a shape. In building up to what we do with D3, we're going to use a "scalable vector graphic," or SVG.

A SVG is a graphic you define with text. Like the commands that define the webpage, the SVG command begins with <`svg`> and ends with </`svg`>.

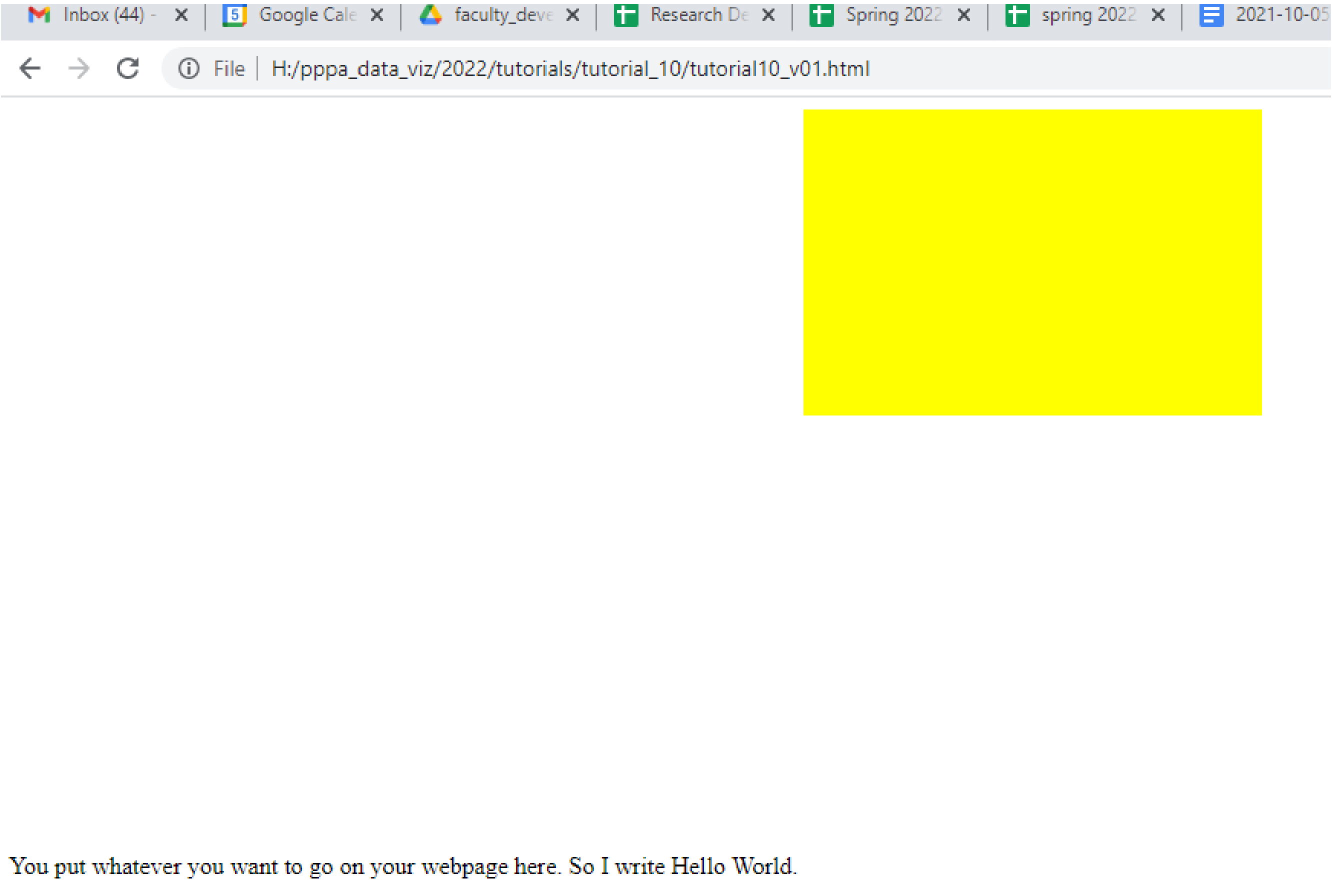
We'll start with a yellow rectangle. Modify your code to add the yellow rectangle:

```
<!DOCTYPE html>

You put whatever you want to go on your webpage here.  So I write Hello World.

<svg width = "500" height = "500">
  <rect x = "0" y = "0" width = "300" height = "200" fill = "yellow"></rect>
</svg>

</html>
```

We are telling the webpage to save space for graphics that is 500 pixels wide and 500 pixels high. Inside there, we want to draw a rectangle (`rect`) that is 300 pixels wide and 200 pixels tall that we fill in yellow.

Re-save, and refresh the file in your browser window.

My file now looks like



You put whatever you want to go on your webpage here. So I write Hello World.

3.3. Adding sections to your webpage

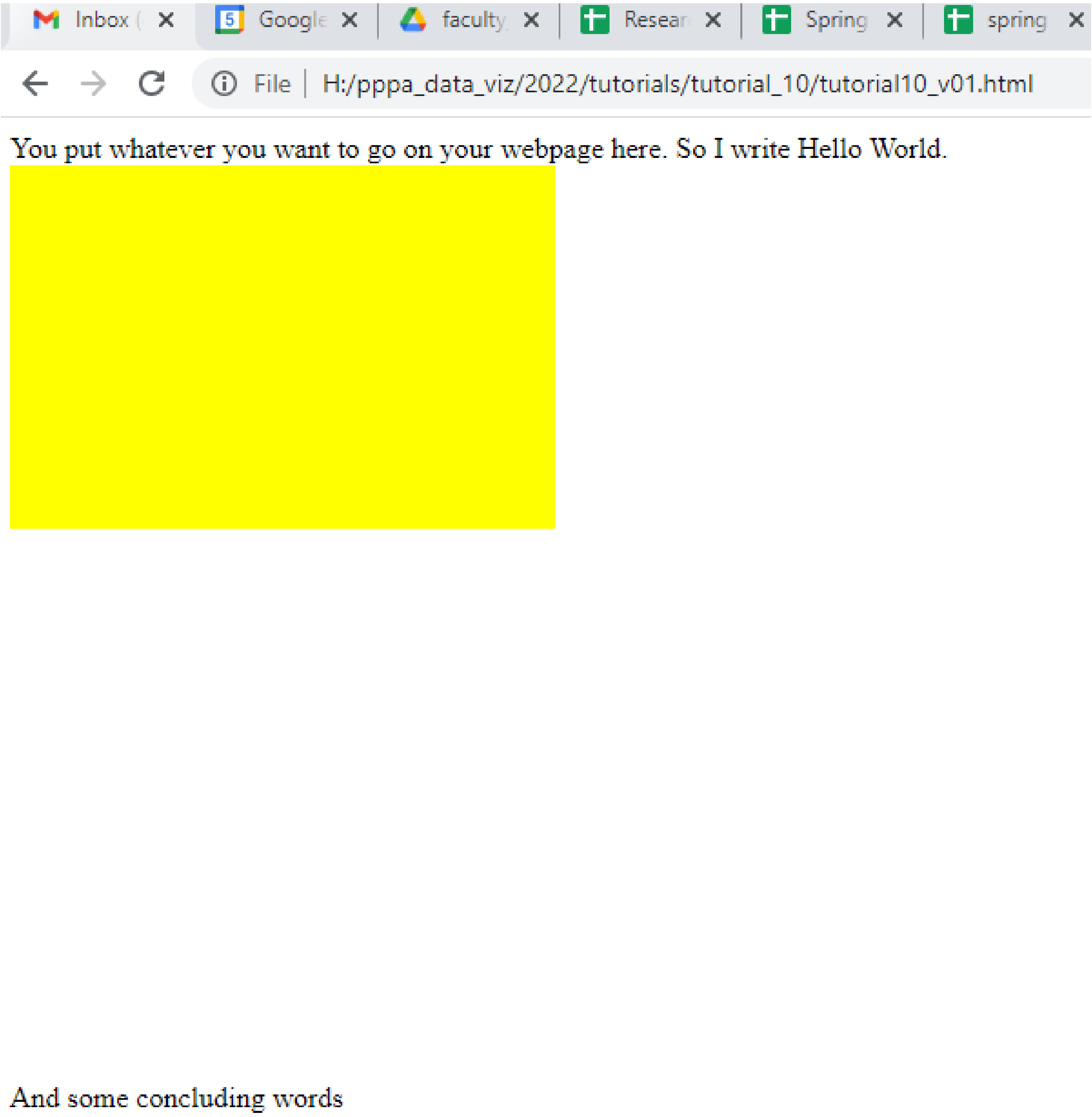
It is frequently useful to divide up your webpage into sections. In this tutorial, we do this using `<div> </div>`. This is useful if you later want to do something systematically to particulary sections.

We name each `div` section with an id phrase of our own choosing. This id goes inside the `div` command, as in `<div id = ‘‘whatever you make up’’> </div>`.

I'm going to add a little more text and divide this simple webpage into three sections using the <div> command. Note that each div section has its own id.

```html
<!DOCTYPE html>

<div id = "intro"> You put whatever you want to go on your webpage here.  So I write Hello World. </div>

<div id = "rectangle.section">
<svg width = "500" height = "500">
  <rect x = "0" y = "0" width = "300" height = "200" fill = "yellow"></rect>
</svg>
</div>

<div id = "last.bit">
  And some concluding words
</div>

</html>
```

When you look at your file (mine is below), you'll see that the page is now ordered differently. The <div> command forces pieces to process in order, so that the yellow rectangle is now in the middle. You can also clearly see how the box of the yellow rectangle is much taller than the rectangle itself (500 vs 200).



H:/pppa_data_viz/2022/tutorials/tutorial_10/tutorial10_v01.html

You put whatever you want to go on your webpage here. So I write Hello World.

And some concluding words

4. Basic Bar Chart in D3

Our goal is to make an interactive graphic. To understand what we're doing, we'll start with a static D3 plot and then move to a interactive one.

4.1. Header code for chart

For the barchart, create a new html document. I save mine as `tutorial10_static_bar_chart_v01.html`.

```html
<!DOCTYPE html>
<meta charset="utf-8">

<!-- Load d3.js -->
<script src="https://d3js.org/d3.v6.js"></script>

<!-- Create a div where the graph will take place -->
<div id="my_dataviz"></div>

</html>
```

This is, so far, a document that would show nothing. We've added commands to

- load the D3 javascript package, so that the html document can interpret the D3 commands: `<script src = ".`
- create a "div" in which our graph will sit: `<div id="my_dataviz"></div>`

We've also added comments. In R, comments start with #. In html, comments go inside `<!-- -->`.

This code is the same regardless of what type of D3 graph you're making – bar, scatter, whatever. You start by telling the webpage that you need to use the d3 javascript library and that you're making a space for your graph.

4.2. Grabbing data

For the graph we're making, we're going to grab data from a website. To understand what the program is doing, open a new browser window and enter `https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/7_OneCatOneNu` `header.csv`

You should see a ten-observation dataset with two variables: County and Value. This is a dataset of military sales by country. Notice that this dataset is already in perfect shape to make a bar graph: we have totals by category, and no extraneous information.

4.3. The simplest bar plot

Now we'll add the text for the body of the chart. I'm pasting the entire code below. If you'd like to copy it directly from the website, I'm using the example from here with very few modifications.

This code has three big parts

A. Setting the dimensions and margins of the graph

B. Appending the svg object we create to the body of the page

C. Parsing the data to create the graphic

This a somewhat backwards order; my intuition would be to draw the graph and then append it to the page, but that is not the order this command uses.

Now let's look a little more into what each part does. I encourage you to look at this explanation and at the code below, back and forth.

A) Setting the dimensions and margins of the graph

  - This portion just sets the size of the box the graph sits in
  - To modify the size of this box, change the width from 460 or the height from 400
  - You can also modify the margins in the very first line

B) Appending the svg object we create to the body of the page

  - The command `d3.select("#my_dataviz)` is telling the page to put the SVG object it's creating in the `div` we created at the top. This reference needs to have the same name as the id of the `div`, or nothing appears.
  - Inside this command, we're putting two things together
  - First the graphic definition with `.append("svg")`
  - And graphic specifics, defined as `g` below
  - If you remove the `.append("svg")` part the graphic goes away
  - If you remove the `.append("g")` part, the y axis goes away

C) Parsing the data to create the graphic

  - The line that begins `d3.csv(...` is a key line – it tells the webpage where to find the data
  - You **cannot** modify this to refer to data on your own computer
  - Why? For security reasons, web browsers do not let you write code that looks into a user's computer (ok, that seems reasonable)
  - I also can't put data on my course website that you can call with this `d3.csv()` command
  - Why? Because my website has some setting that doesn't allow programs like this to read data directly
  - However, a commonly used coding website called github does that settings that allow you to read data, so I post some data for today's tutorial there

  a) X axis

    - here we define how we want the X axis to look
    - this part of the command tells D3 that we want country for the x axis: `.domain(data.map(d => d.C`
    - note the negative 45 degree angle for the labels

  b) Y axis

    - Here we define the y axis
    - Note that we don't mention a specific variable
    - And we hard-code the maximum value of 13,000

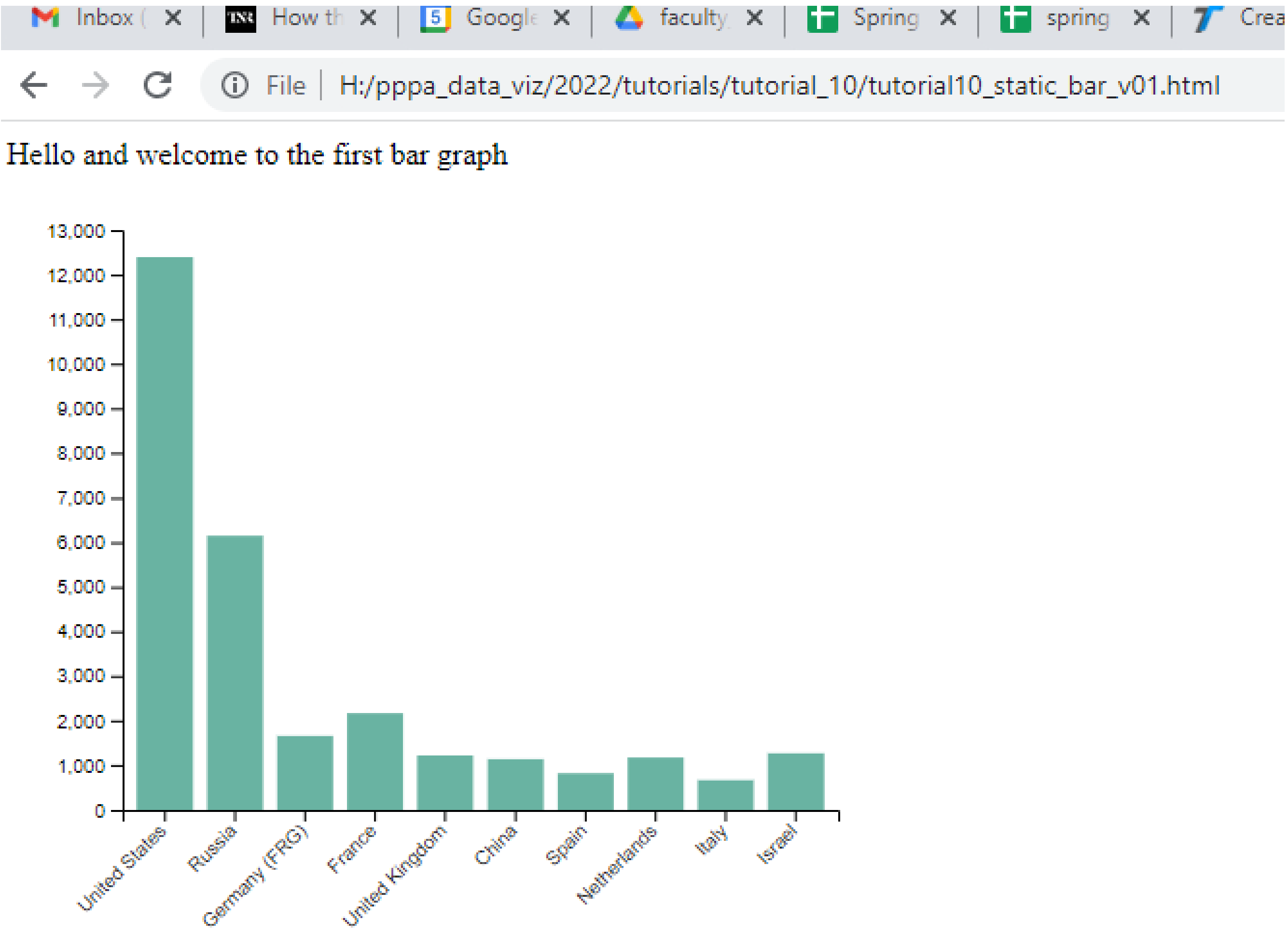
c) Bars

- This section draws a bunch of rectangles (`.join("rect")`)
- And put them together into something called `mybar`, though I'm pretty sure this name does not matter
- Note that we define the x and y variables in this piece of code
- The line `.attr("fill", "#69b3a2")` fills in the bars with a particular hexcode color – this should look vaguely familiar

```
<!DOCTYPE html>
<meta charset="utf-8">

<!-- Load d3.js -->
<script src="https://d3js.org/d3.v6.js"></script>

<!-- Make a bit of a intro header -->
<div id = "intro.bit">
  Hello and welcome to the first bar graph
</div>

<!-- Create a div where the graph will take place -->
<div id="my_dataviz"></div>

<!-- start doing the d3 processing -->
<script>

// A. set the dimensions and margins of the graph
const margin = {top: 30, right: 30, bottom: 70, left: 60},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// B. append the svg object to the body of the page
const svg = d3.select("#my_dataviz")
  .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", ‘translate(${margin.left},${margin.top})‘);

// C. Parse the Data
d3.csv("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/7_OneCatOneNum_header.csv").then(

  function(data) {

  // C.1. X axis
  const x = d3.scaleBand()
    .range([ 0, width ])
    .domain(data.map(d => d.Country))
    .padding(0.2);
  svg.append("g")
    .attr("transform", ‘translate(0, ${height})‘)
    .call(d3.axisBottom(x))
    .selectAll("text")
      .attr("transform", "translate(-10,0)rotate(-45)")
      .style("text-anchor", "end");

  // C.2. Add Y axis
  const y = d3.scaleLinear()
    .domain([0, 13000])
    .range([ height, 0]);
  svg.append("g")
    .call(d3.axisLeft(y));

  // C.3. Bars
  svg.selectAll("mybar")
    .data(data)
    .join("rect")
      .attr("x", d => x(d.Country))
      .attr("y", d => y(d.Value))
      .attr("width", x.bandwidth())
      .attr("height", d => height - y(d.Value))
      .attr("fill", "#69b3a2")

})

</script>

</html>
```

Save this document and open it in a browse window as we did before.

My output looks like



H:/pppa_data_viz/2022/tutorials/tutorial_10/tutorial10_static_bar_v01.html

Hello and welcome to the first bar graph

Now that you've made a static bar chart, you're ready to head off to the world of interactivity.

5. Interactive Bar Chart

In this section, I hoped to have us do an interactive bar chart based on data from the American Community Survey. But despite working on this for quite a bit, I have not created a good example! Instead, we'll work through a simpler example of an interactive bar.

5.1. Code for Interactivity

The code below creates an interactive version of the type of static bar chart we just made.[1]

We are introducing buttons to select the variable for which you want to see the bar chart (variable 1 or variable 2). Intuitionally, the code for these buttons reports to the rest of the program which input it should use.

[1] This code is taken verbatim from the D3 graph gallery **with version 6 turned on**. I am very appreciative for this great example.

In addition, this revised code also has a new section that updates the bars based on user input – note that d3 refers to the user input as `d[selectedVar]`.

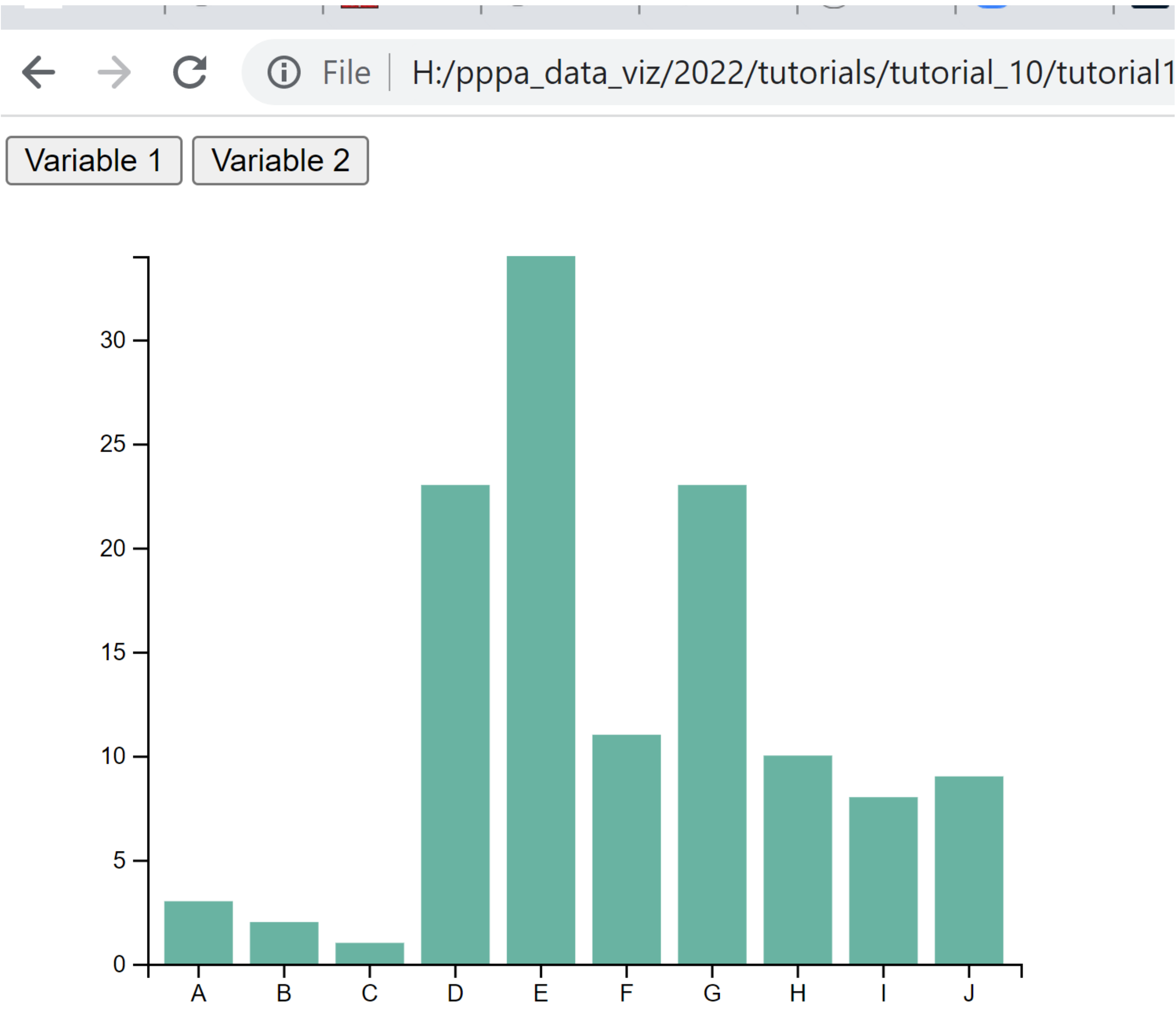
Copy this code into a document in atom that you save with an .html extension. Then open the code in your browser, as you did above.

A word of warning from last semester. For some students, copying the pdf changed quotes and made the code fail to turn. You may wish to copy this code directly from where I got the example (see the D3 graph gallery **with version 6 turned on**).

```
<!DOCTYPE html>
<meta charset="utf-8">

<!-- Load d3.js -->
<script src="https://d3js.org/d3.v6.js"></script>

<!-- Add 2 buttons -->
<button onclick="update(’var1’)">Variable 1</button>
<button onclick="update(’var2’)">Variable 2</button>

<!-- Create a div where the graph will take place -->
<div id="my_dataviz"></div>

<script>

// set the dimensions and margins of the graph
const margin = {top: 30, right: 30, bottom: 70, left: 60},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// append the svg object to the body of the page
const svg = d3.select("#my_dataviz")
  .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", ‘translate(${margin.left},${margin.top})‘);

// Initialize the X axis
const x = d3.scaleBand()
  .range([ 0, width ])
  .padding(0.2);
const xAxis = svg.append("g")
  .attr("transform", ‘translate(0,${height})‘);

// Initialize the Y axis
const y = d3.scaleLinear()
  .range([ height, 0]);
const yAxis = svg.append("g")
  .attr("class", "myYaxis");


// A function that create / update the plot for a given variable:
function update(selectedVar) {

  // Parse the Data
  d3.csv("https://raw.githubusercontent.com/holtzy/D3-graph-gallery/master/DATA/
    barplot_change_data.csv").then( function(data) {

    // X axis
    x.domain(data.map(d => d.group));
    xAxis.transition().duration(1000).call(d3.axisBottom(x));

    // Add Y axis
    y.domain([0, d3.max(data, d => +d[selectedVar]) ]);
    yAxis.transition().duration(1000).call(d3.axisLeft(y));

    // variable u: map data to existing bars
    const u = svg.selectAll("rect")
      .data(data)

    // update bars
    u.join("rect")
      .transition()
      .duration(1000)
        .attr("x", d => x(d.group))
        .attr("y", d => y(d[selectedVar]))
        .attr("width", x.bandwidth())
        .attr("height", d => height - y(d[selectedVar]))
        .attr("fill", "#69b3a2")
  })

}

// Initialize plot
update('var1')

</script>

</html>
```

5.1.1. Output for Interactivity

My output looks like



—

To actually make substantive changes to this set-up, you need to know quite a bit more d3 that I know and that I can teach you in one tutorial. Hopefully, however, this tutorial has given you the flavor of how you might create interactive charts, and an experience with coding for charts that is not R.

6. Homework

For this week's homework, please turn in one pdf document that has the following:

- Written answers to today's homework questions with pictures or screenshots as requested
- Your html code

1. Make some minor modification to the text in one of the webpages so that I know that you've actually tried this.

2. Change the color of the bars in the static bar chart from Section **??** to a color of your pleasure. To do so, create a new .html document, using the code from Section **??**, and modify as needed. For your answer to this question, please write which part of code you modified, and take a screenshot of your new graph.