CSCI 4210 U – Information Visualization
Lab 4. Filtering

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

PURPOSE

Enhance a static visualization by implementing interactive filtering controls. Exercise implementing dynamic behaviour with D3's enter and exit selections.

TASKS

In this lab you are given a static visualization of the Diamonds dataset that depicts the relation between diamond price and carat in a scatterplot. **You are asked to make this visualization interactive by implementing filtering functionality.**

Open index.html with your editor of choice. Then use python to serve the page:

```
cd lab4folder
python3 -m http.server 8080
```

Now access the page on http://localhost:8080.

Note the structure of the document. The left panel is a div with id="filters". Inside this div you will find a range slider for the continuous attribute price, and a set of checkboxes for the ordinal attribute cut. Price is mapped to vertical position in the scatterplot, while cut is mapped to color.

For the purposes of this lab, you do not need to edit the CSS or the HTML structure of the page.

1. **Upon loading the data, set the correct minimum and maximum values for the range slider.**

The range slider is initialized with the function.slider(), which receives an object where the min and max are defined, as well as the initial values of the slider, and any event listeners. This function is currently being called before the data is loaded. Move it to the correct place and update the slider values to match the price column in data.

Remember you can use d3.min, d3.max, and d3.extent to get an array's extrema. Array's map function may be useful.

2. Filter the data and update the display whenever a slide event is triggered.

You can choose among many filtering strategies. A basic strategy is to keep two arrays: one with all records and other with the filtered data. When a filter changes, the second array is overridden with the result of calling Array.filter on the first.

Once you have the filtered array, bind it to your D3 selection. Make sure you remove extra elements, which are stored in the exit selection. And create new elements, which are accessible under the enter selection.

If you need to refresh your understanding of D3 selections, check out this jsfiddle, which makes use of D3 enter, exit, and update; or read this concise explanation by D3's author Mike Bostock.

3. Filter the data and update the display when a checkbox event is triggered.

The filters (price and carat) should work together. That is, if an unchecked box is checked, then only the records that correspond to that box AND lie within the currently selected price range should enter the display. Likewise, when the slider is changed,

data records should not enter the display if they have a cut value that is not selected in the cut filter.

4. Update the scale and axis.

The price axis should respond to changes in the filter. In D3, axes are created from scales:

```
var yAxis = d3.axisLeft(yScale);
```

This is very convenient because the yScale is probably already defined (it was needed to determine the vertical position of points).
The axis is then drawn by creating an SVG group and calling the axis on the corresponding selection.

```
d3.select('svg')
          .append("g")
          .attr("transform", "translate(40,0)")
          .call(yAxis);
```

In the code above, yAxis will “create itself” inside the SVG group (“g element”). It knows the proper axis values from the scale we passed at creation time.

In order to update the axis, you just need to recreate the scale with the new data, then empty the SVG group that wraps the old axis, and repeat the axis creation calls.

SUBMISSION

Submit index.html to the lab dropbox on Blackboard.