

# CSE 564 Visualization and Visual Analytics - Lab1 Report

Student: Bo Cao

SBU ID: 112130213

Email: [bo.cao.1@stonybrook.edu](mailto:bo.cao.1@stonybrook.edu) or [boccao@cs.stonybrook.edu](mailto:boccao@cs.stonybrook.edu)

## Demo

This code is live on

<https://blockbuilder.org/BryanBo-Cao/1e2f511ee3bbdae742777dc73cf7a441>

PLEASE switch to side-by-side mode to see the whole functionality of this code instead of using fullscreen.

Video: <https://youtu.be/0LYa9eJWVlg>

## File Structure

All files include **index.html**, **College.csv**, **compute\_min\_spanning\_tree.py**, **minimum\_spanning\_tree\_mtx.json** where **1) index.html** is the main file to run and all the d3 code is in this file; **2) College.csv** is the dataset downloaded from the college dataset from <https://vincentarelbundock.github.io/Rdatasets/datasets.html>, the original dataset has College 777 data points, 18 dimensions; **3) compute\_min\_spanning\_tree.py** is the python code to compute the minimum spanning tree for the first 70 data points and save the data as **4) minimum\_spanning\_tree\_mtx.json**, then index.html visualize it in force-directed layout graph. Note that in the force-directed layout graph, the distances between two nodes are the euclidean distance using all the attribute.

## Code

Bootstrap is used as the base of css. The code first creates a canvas, then it creates barchart\_svg, piechart\_svg and force\_svg to display each kind of chart separately.

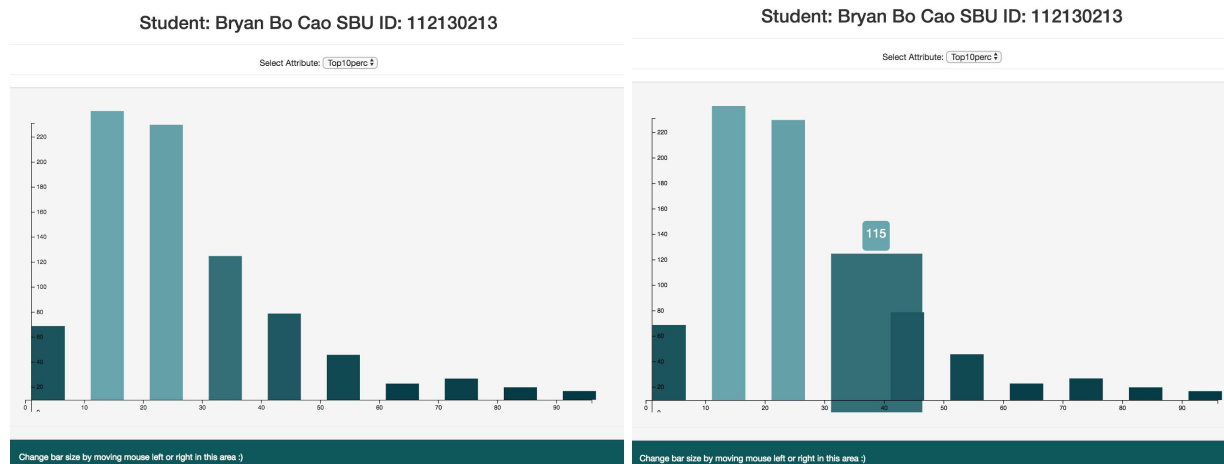
Barchart\_svg is initialized by `init_barchart(display_data_values)`, it computes both x and y scale and calculates bar width and height. And then transform it by

```

bar_h = function(d) { return svg_h - y_scale(d.length)};
bar_c = function(d) { return color_scale(d.length)}, // bar color
bar_t = function(d) { return "translate(" + x_scale(d.x0) + "," + y_scale(d.length) + ")";}; //
bar transform

```

Each rectangle is appended to barchart with two events listener -- mouseover and mouseout. Mouseover is used to enlarge the corresponding rectangle and show the content on top of the bar while mouseover is used to define the action to make the rectangle smaller.

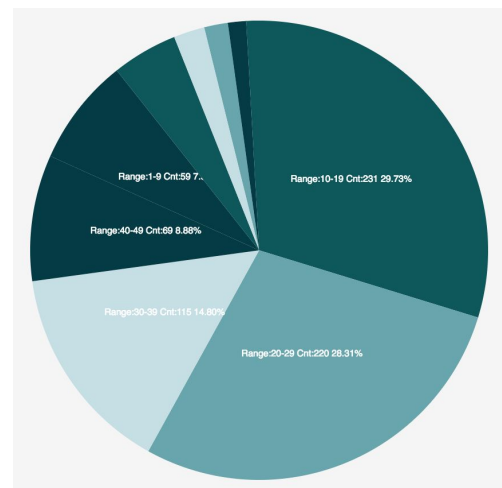


Similarly, for piechart, scaled data is stored in display\_data4piechart, and bound by

```

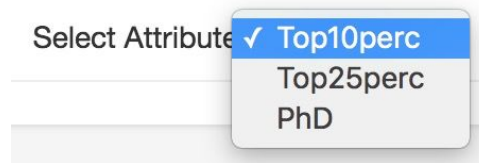
var group = piechart_svg.append("g")
    .attr("transform", "translate(550, 300)");
var pie = d3.pie().value(function(d) { return d; });
var arc = d3.arc().innerRadius(0).outerRadius(radius);
var arc_group = group.selectAll(".arc")
    .data(pie(display_data4piechart))
    .enter()
    .append("g")
    .attr("class", "arc");

```



Whenever a mouseclick event is happened, the next type of chart will be displayed while the other two will be hidden, which is done by the function change\_chart().

To select another variable, you can simply click on the dropdown menu and select one of other variables. The dropdown menu is shown below



When a different variable is selected, it will call the functions of `update_barchart()`, `update_piechart()`. Note that the minimum spanning tree is computed based on all attributes that I use, therefore changing attribute will not change the force directed layout in my code.

To decrease or increase the size of bar, a user needs to move the mouse over the `<div id=change_bar_size_div>` area, shown below

Change bar size by moving mouse left or right in this area :)

Then it will listen to `onmousemove` event by

```
d3.select("#change_bar_size_div")
    .on('onmousemove', check_lr_movement);
```

In the function `check_lr_movement()`, this effect is done by comparing the mouse position between the current and the previous one, then change the bar width based on the previous size by

```
if (pre_mouse_x_pos < event.clientX) {
    // move right to make bars thicker
    curr_bar_w++;
} else {
    // move left to make bars thinner
    curr_bar_w--;
}
```

Dataset is loaded by `d3.csv("College.csv", function(data)` and minimum spanning tree matrix is loaded by `d3.json("minimum_spanning_tree_mtx.json")`.

Note that D3 Version 4 is used for visualization, Python 3.7 is used to run `compute_min_spanning_tree.py`.