# CSE 564 - Visualization Mini Project 1

Name: Dhruva Gaidhani
SBU id: 111971181

A. Introduction:

A CSV file containing data about forest fires in Canada[1], has been taken for this project from Kaggle to perform data visualization on it. The data file is present as forestfires.csv in the src folder along with a forestfires.names txt file with information about the attributes in the dataset. The data has 517 rows with no null values and 12 columns out of which 8 have been visualized.

This report summarizes the implementation of the following:

I. Using the JavaScript library for building a user interface built up from the sample files created while initializing it.
II. Using D3.js for data driven visualization with mouse events like click, mouse-in and mouse-out.
III. The interface is minimalist with self-explanatory layout built using HTML and CSS.

B. Overview:

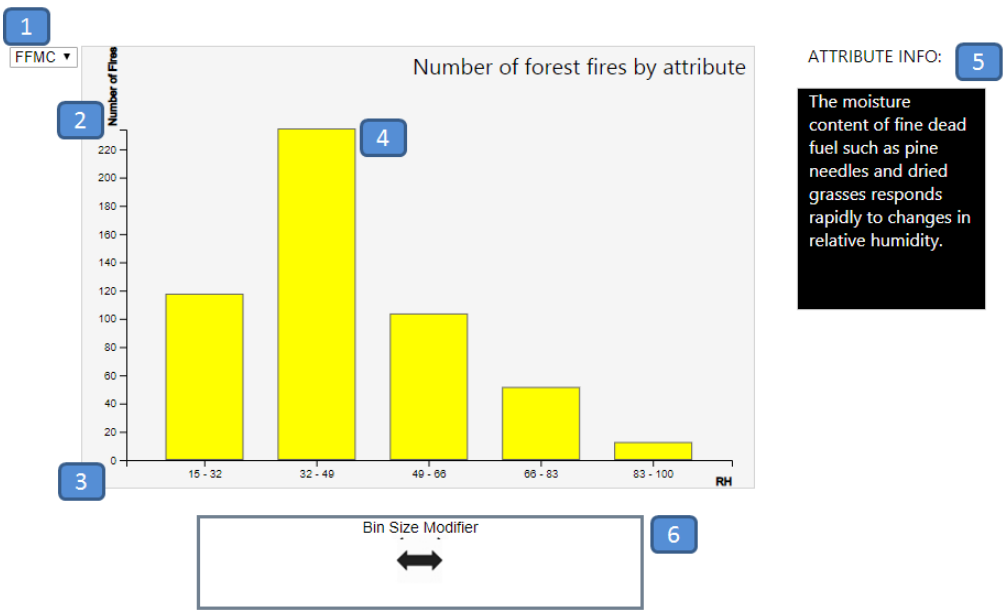


*Fig1. Layout of the interface upon initial loading.*

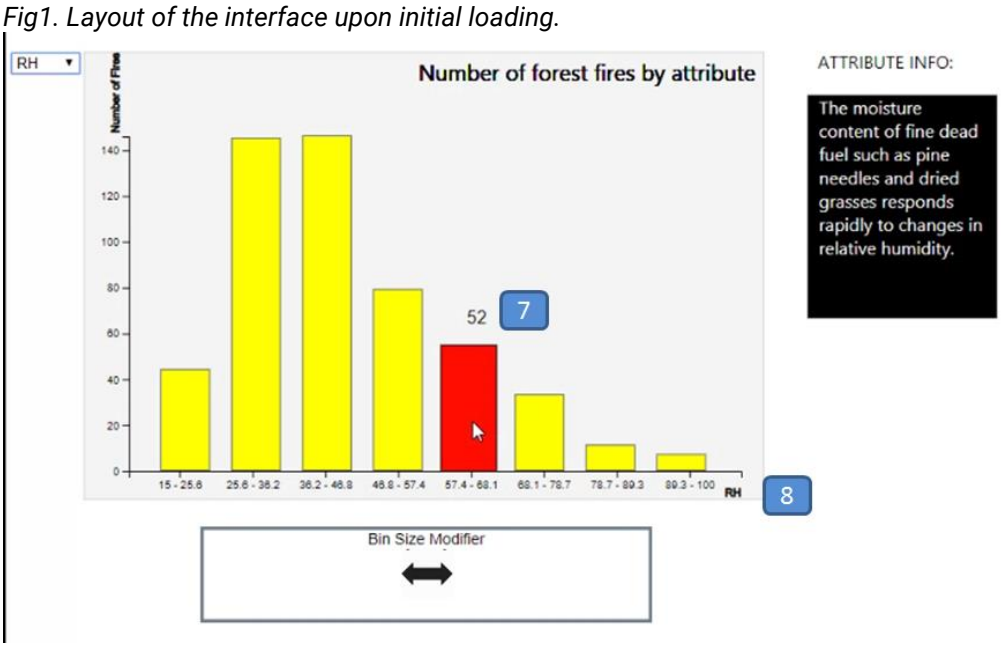


*Fig2. Handling Mouse-events on screen one with changed attributes.*

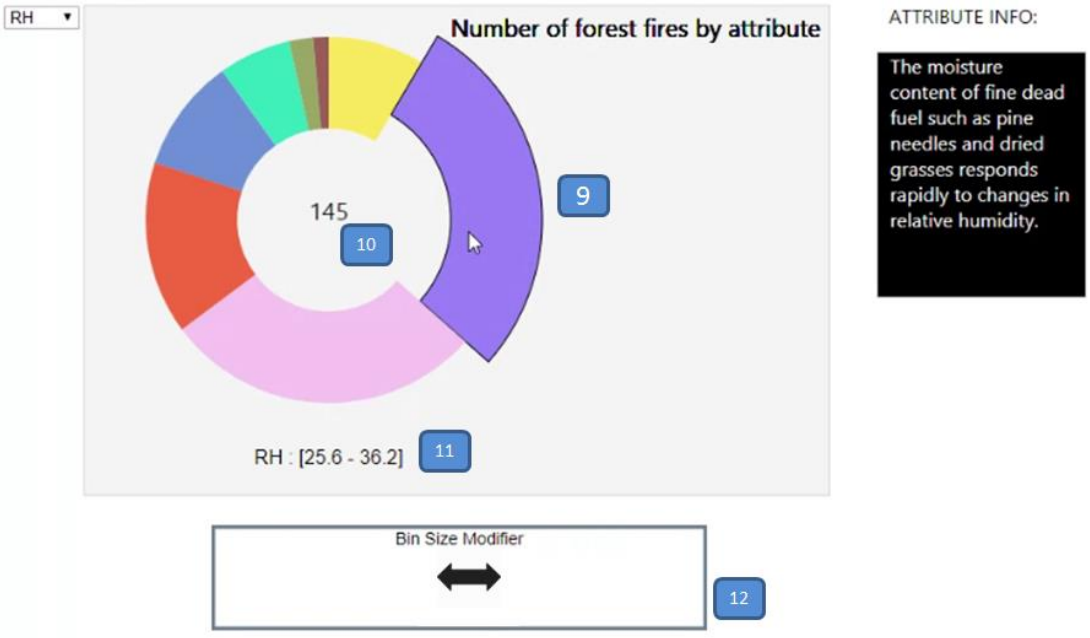


*Fig3. Layout upon mouse click changes the visualization to a pie chart.*

B. Component analysis overview:

1. Menu for displayed attribute selection
2. Y-axis displaying the magnitude of the attribute which sizes dynamically.
3. Origin created at the intersection of the X-axis and the Y-axis.
4. Bar-charts displaying visually the magnitude of the selected attrbutes
5. Info block that displays more data about the attribute.
6. Bin-size modifier that changes the input of number of bins to the plot function upon mouse hover
7. Mouse hover effect on the bar-chart
8. Dynamically changing X-axis
9. Pie-chart with mouse hover movement
10. Magnitude of the attribute selected
11. Range display for the bin being focussed on
12. Bin size modifier works for the pie-chart as well.

C. Task-wise description:

1. Pick a variable and bin it into a fixed range (equi-width) of your choice:

Code snippet:

```
switch(attribute){
    case "FFMC":
        bin_width = Math.floor((fire_max.FFMC - fire_min.FFMC) / bin_number * 100) / 100;
        lowerBound[0] = fire_min.FFMC;
        upperBound[bin_number-1] = fire_max.FFMC;
```

The bar graph is built by default on the FFMC attribute with bin-size as 5. The bin-size is given as input based on the slider value and the width of the bar-graph is decided by the range and the scale as given by the formula in the question slides.

2. Create a bar chart of the variable you picked in 1:

Code snippet:

```
rects = svg.selectAll("rect")
    .data(bar_data.count)
    .enter()
    .append("rect")
    .attr("class", "bar")
    .attr("id", "bar_chart")
    .attr("x", function(d, i){
        return 46 + (i * bar_w * 1.019) + (bar_padding / 1.5);})
    .attr("y", function(d, i){
        return 14.5 + 0.9 * svg_height - bar_data.scale(d);})
    .attr("height", function(d, i){
        return bar_data.scale(d);})
    .attr("width", bar_w - bar_padding);
```

D3 library is used to select the svg element and create objects of type rectangle with the width and height as supplied by the bin_data variable which is calculated in the visualize data function.

3. Using a menu, allow users to select a new variable and update chart:

Code snippet:

```
drop_down_menu.append("select")
    .attr("class","menu")
    .selectAll("option")
    .data(options)
    .enter()
    .append("option")
    .attr("value", function(d){
        return d;})
    .text(function(d){
        return d;})
```

A normal drop down menu is created that chooses the attributed and passes it to the plot function which changes the data that is being plotted.

4. Only on mouse-over display the value of the bar on top of the bar:
5. On mouse-over make the bar wider and higher to focus on it

Code Snippet:

```
rects.on("mouseover", function(d, i){

    d3.select(this)
        .transition()
        .attr("x", 46 + (i * bar_w * 1.019) + (bar_padding / 1.5) - 4)
        .attr("width", 8 + bar_w - bar_padding)
        .attr("y", 14.5 + 0.9 * svg_height - bar_data.scale(d) - 7)
        .attr("height", bar_data.scale(d) + 8);
        // Add value above a bar
```

The bar graph rect object is midified on mouseover to increase the width and the height of the bar to focus on it. On mouse over another text object is appended to the svg which shows the exact value of the bar graph being looked at.

6. on mouse-click transform the bar chart into a pie chart (and back)
Code Snippet:

```
// Event 1: Click
rects.on("click", function(d, i){
    svg.selectAll("#bar_label").remove();
    svg.selectAll("#bar_chart").remove();
    svg.selectAll("#axis").remove();
    svg.selectAll("info").remove();
    //console.log(fire_data);
    //console.log(bin_data);
    //console.log(bar_data);
    visualizeData("pie-chart", fire_data, bar_data.bin_number, bar_data.attribute);
```

A mouse click event calls the visualizeData function which removes all the bar graph elements and appends the pie-chart elements to the svg. A similar function in the pie-chart methods allows the required change back. The pie-chart is chosen to be composed of arcs so that the data can be displayed easily in the center of the graph and thus not be obscured by the color of the bar-graph itself. It also reduces unnecessary complexity involved around finding the center of each of the arcs of a solid pie-chart.

7. Mouse moves left (right) should decrease (increase) bin width/size

```
// Mouse over handler for bin size mod
d3.selectAll(".touch_controller")
    .on("mouseover", function(d,i){
        console.log(i);
        visualizeData(bin_data.chart_mode, fire_data, i+1, bin_data.attribute);
    });
```

The bin size modifier is an area of invisible div elements grouped together under the slider class. This javascript part of the code iterates through the elements of this group and supplies the iterator index to the visualizeData function which in turn takes down previously built elements and builds a different graph on top of the elements.

**Extra Credit** – 1 -- An additional 10 pts for elegant implementation/function

The code has been written in the React framework.

References:


[1] P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. *Available at:* http://www.dsi.uminho.pt/~pcortez/fires.pdf

[2] https://bl.ocks.org/shashank2104/d7051d80e43098bf9a48e9b6d3e10e73

[3] http://bl.ocks.org/miklobit/6421a0e3dea38e3bbdb1

[4] https://www.themepunch.com/faq/mouse-hovers-layer-content/

[5] http://bl.ocks.org/andreaskoller/7695784

[6] https://stackoverflow.com/questions/27238845/manipulate-mouseover-hit-areain-d3-js

[7] https://javascript.info/mousemove-mouseover-mouseout-mouseenter-mouseleave

[8] https://bl.ocks.org/markarios/6b8b91e0c4ce77b4942659aace8d2d6c

[9] http://bl.ocks.org/phil-pedruco/9032348

[10] https://github.com/vijayst/d3bar