Baseball Team Characteristics

Visualization Assignment – 1

February 14, 2017

The d3 based visual interface displays the distribution of baseball players based on various traits like their height, weight, Averages and Home Runs. Following details explain how the various requirements were implemented in this interface.

The interface is built on top of the data available in baseball_data.csv file. The file contains details about the height, weight, average and home runs for a group of baseball players. The file script.js contains the d3 based java script code that renders the page. A high level structure of the script file is as given below :

```
//function to read the Data from the csv file.
function getData() {
}

//function to filter and format/modify the data as per the requirement.
//arg1 -Selected attribute from drop down, arg2 -- binSize , arg3 - formated data from csv
function createHist(arg1, arg2, arg3) {
}

//function to render the bar graphs based on the formatted data.
function renderBar(csvData) {

}

////function to render the pie chart based on the formatted data.
function renderPie(csvData) {
}

//function  to implement the choice list in the page.
function set_attribute(d) {
}

//function to adjust the bin width/size based on mouse movement.
function adjustBin(event){}
```

Requirements

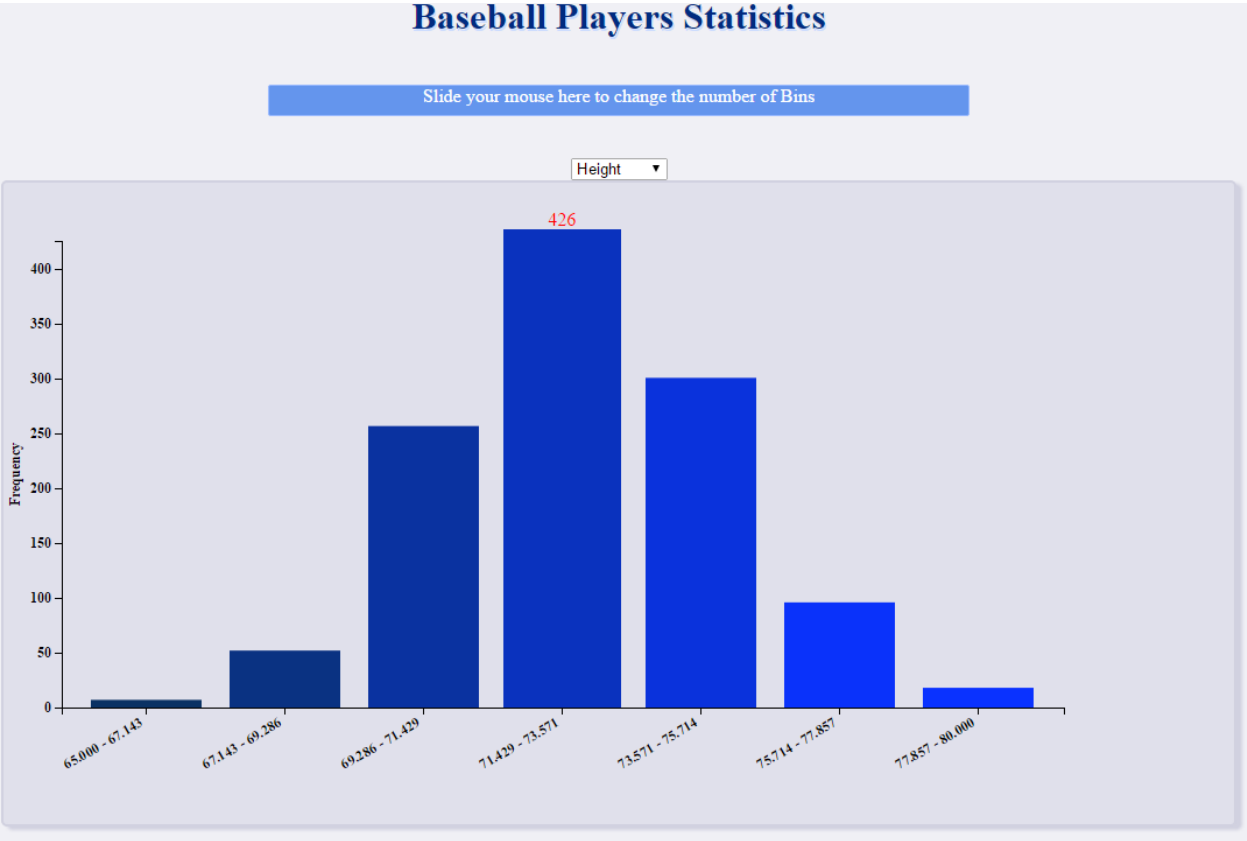
1. Pick a variable and bin it into a fixed range (equi-width) of your choice

By default, the graph is rendered showing a plot of groups of heights of players against the number of players falling in each group.
Data is read from the csv file using the d3.csv API. The data retrieved is formatted and filtered based on the selected attribute (height, weight etc.) and a new data structure is created which contains the column data from the chosen attribute alone. That is ,if the chosen attribute is Height, then the variable 'histVar' in createHist will contain the heights data alone from the csv file. D3 histogram API is used to create the bins based on this data and the binSize as specified. The data is then transferred into 'xyData' which transforms the data into frequency-group plot with x coordinate pointing to the bins created and the y coordinate pointing to the size of the group.

```
function getData() {
    d3.csv("baseball_data.csv", function(error, data) {
        createHist(selection, binSize, data);
    })
}

//function to filter and format/modify the data as per the requirement.
 //arg1 -Selected attribute from drop down, arg2 -- binSize , arg3 - data from csv file
 function createHist(arg1, arg2, arg3) {
    var histVar = [];
    arg3.forEach(function(d) {
        histVar.push(+d[arg1]);
    })
    cloneData = arg3;
    var histogram = d3.layout.histogram()
        .bins(arg2)
        (histVar);
    var xyData = [];

    histogram.forEach(function(d) {
        xyData.push({
            x: "" + (d.x).toFixed(3) + " - " + (d.x + d.dx).toFixed(3),
            y: d.y
        })
    });
    if (ischoicePie) {
        renderPie(xyData);
    } else {
        renderBar(xyData);
    }
}
```

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

   Based on the 'xyData' created in the previous step a bar chart is rendered. The x-axis uses an ordinal scale displaying the groups and y- axis uses a linear scale showing the count . The number of groups in the bar chart are created based on the input as given by the user.



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

   A choice list is displayed on top of the Bar chart with 4 options – Height, Weight , Average and Homerun. Based on the change in choice list value, the graph is re-rendered with the group/bin values updated. The histogram (and hence the bins) are recreated by calling the createHist method as described earlier.

```
function set_attribute(choice) {
        selection = choice;
   d3.selectAll("svg")
     .remove()
   createHist(d, binSize, cloneData);
 }
```

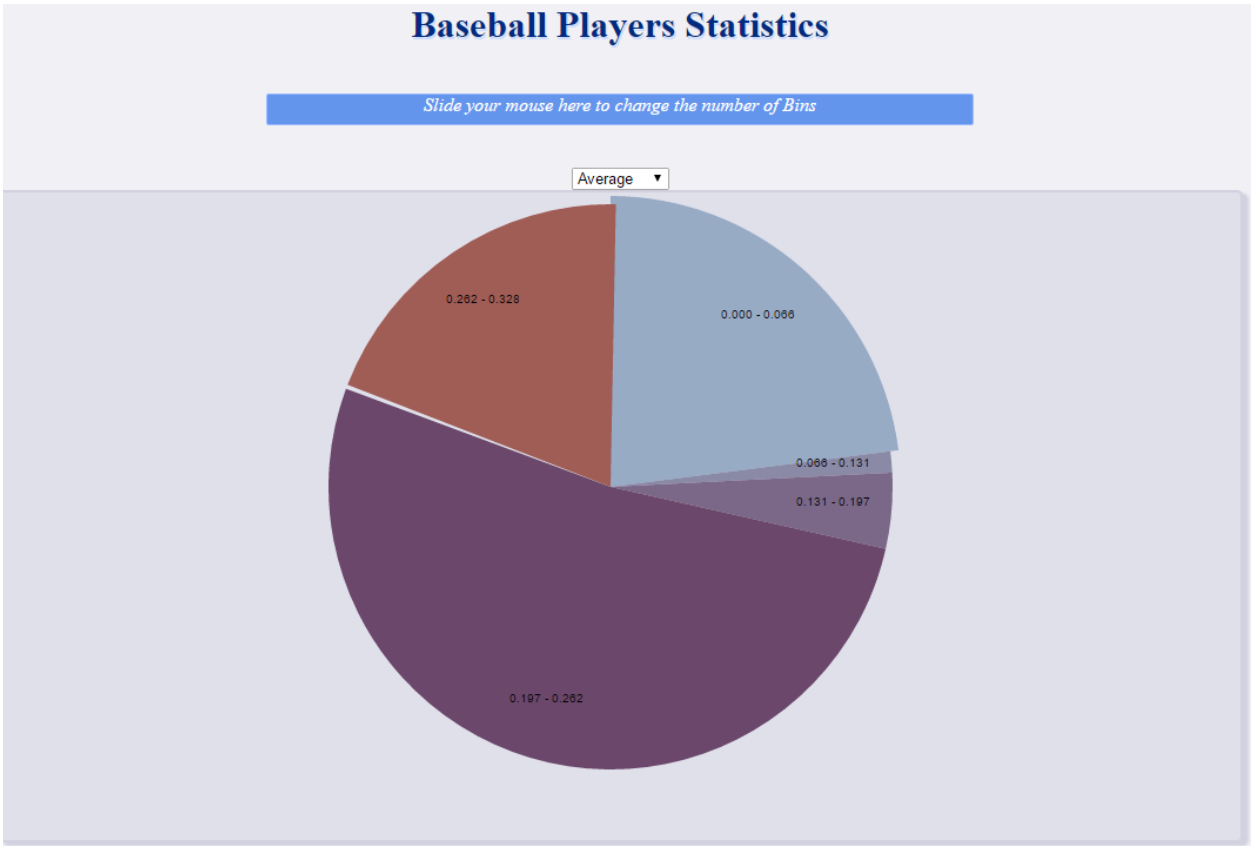
4. only on mouse-over display the value of the bar on top of the bar.
   and

5. on mouse-over make the bar wider and higher to focus on it
   Both these functionalities are achieved by identifying each portion of the bar graph and overriding the mouseout and mouseover methods. Upon mouse over the bars are highlighted by increasing the width and height of the bar and also, the text (value) is shown on top of the bar.

```
 .on("mouseover", function(d, i) {
        d3.select(this)
          .attr("width", xScale.rangeBand() + 6)
          .attr("x", function(d) {
            return xScale(d.x) - 3;
          })
          .attr("y", function(d) {
            return yScale(d.y) - 10;
          })
          .attr("height", function(d) {
            return height + 10 - yScale(d.y)
          })
        d3.select("#text" + i)
          .style("visibility", "visible")
    })   .on("mouseout", function(d, i) {
        d3.select(this)
          .attr("width", xScale.rangeBand())
          .attr("x", function(d) {
            return xScale(d.x);
          })
          .attr("y", function(d) {
            return yScale(d.y);
          })
          .attr("height", function(d) {
            return height - yScale(d.y)
          })
        d3.select("#text" + i)
          .style("visibility", "hidden")
    })
```

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

   The same data that was used to create the bar graph can be used to render the pie chart. The 'xyData' created by the createHist method is used to render the pie chart upon mouse click on the bar graph. A global variable 'isChoicePie' is used to persist the display, and the value of this variable is toggled only upon mouse click on the bar chart or the pie chart. The pie chart is rendered using the d3.svg.arc() and d3.layout.pie() methods. Upon mouse hover the arc that is chosen is highlighted.



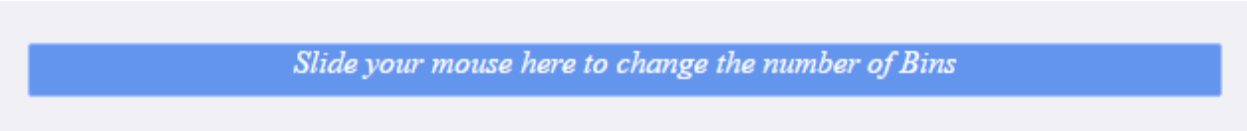

```
  var hoverArc = d3.svg.arc()
    .outerRadius(radius-3)
    .innerRadius(0);

var labelArc = d3.svg.arc()
    .outerRadius(radius - 60)
    .innerRadius(radius - 60);
var pie = d3.layout.pie()
    .sort(null)
    .value(function(d) {
       return d.y;
    });
var g = svg.selectAll(".arc")
    .data(pie(csvData))
    .enter().append("g")
    .attr("class", "arc");

g.append("path")
    .attr("d", arc)
                  .attr("id", function(d, i) {
       return "path" + i;
    })
```

```
.style("fill", function(d) {
   return color(d.data.y);
})
.on("click", function(d) {
   ischoicePie = false;
   renderBar(csvData);
})
                .on("mouseover", function(d, i) {
   d3.select(this)
                        .attr("d",hoverArc)
})
.on("mouseout", function(d, i) {
   d3.select(this)
                       .attr("d",arc)
})
.transition()
.ease(d3.easeLinear)
.duration(2000)
.attrTween("d", pieTween);
```

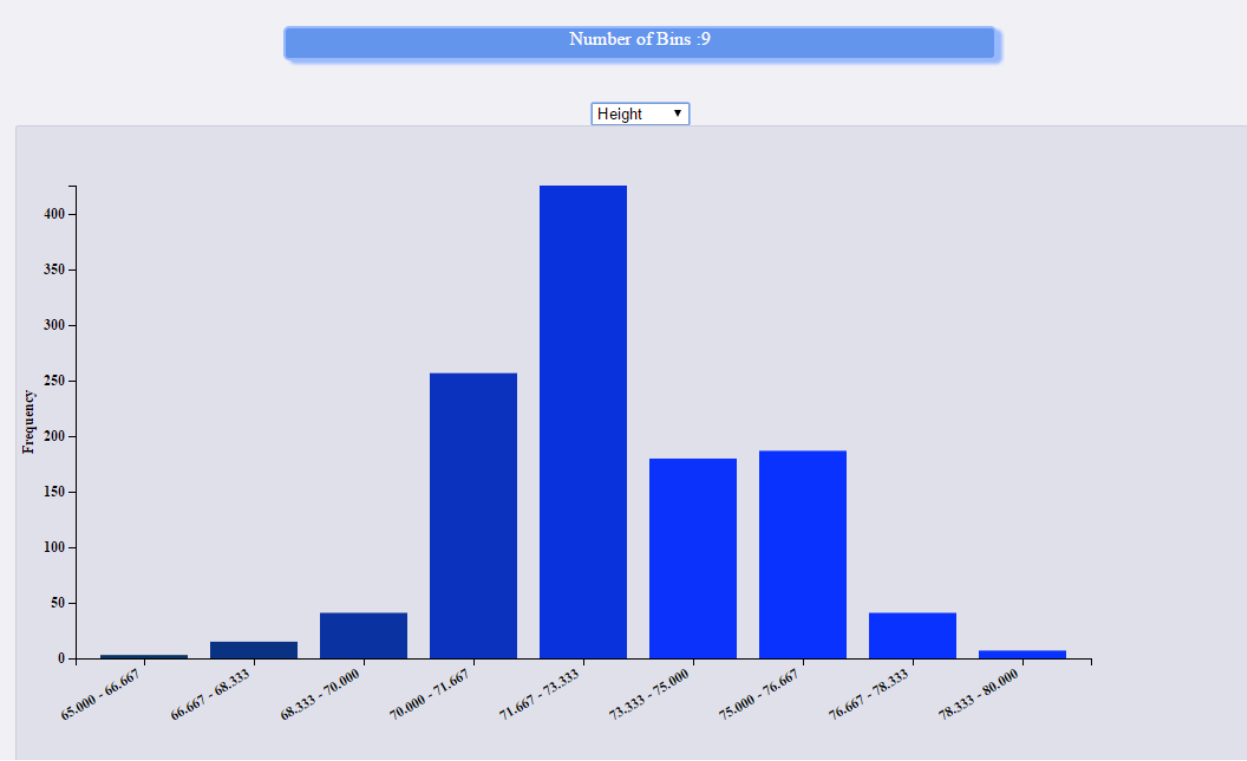
7. mouse moves left (right) should decrease (increase) bin width/size.

   A slider area is added on top of the page with which the user will be able to change adjust the bar size and width displayed on the page. Essentially this parameter controls the number of bins/groups created. The bin variable value is calculated out of a mathematical expression based on the x coordinate of the current position. So moving left or right over the area will result in re-computing the bin value.

   Once the bin size is recomputed , the current svg is removed from the view and the createHist method as mentioned earlier is invoked with the updated bin size value. Based on the value of the global variable ‘isChoicePie’ the bar graph or the pie chart is rendered with the updated bin size value.



Upon mouse move towards left the bins size/width decreases (i.e the number of bins increases)



Upon Mouse move towards right, the bin size/width increases (i.e the number of bins decreases)