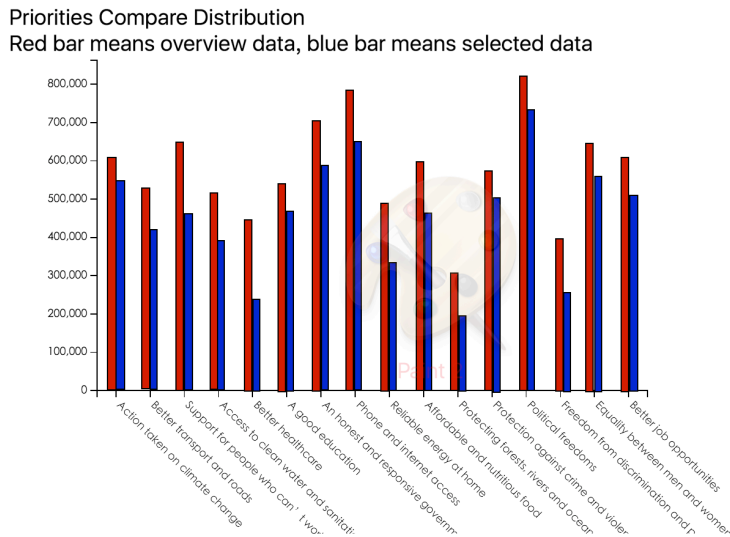


HW4 Analysis Design

1. Task 4a

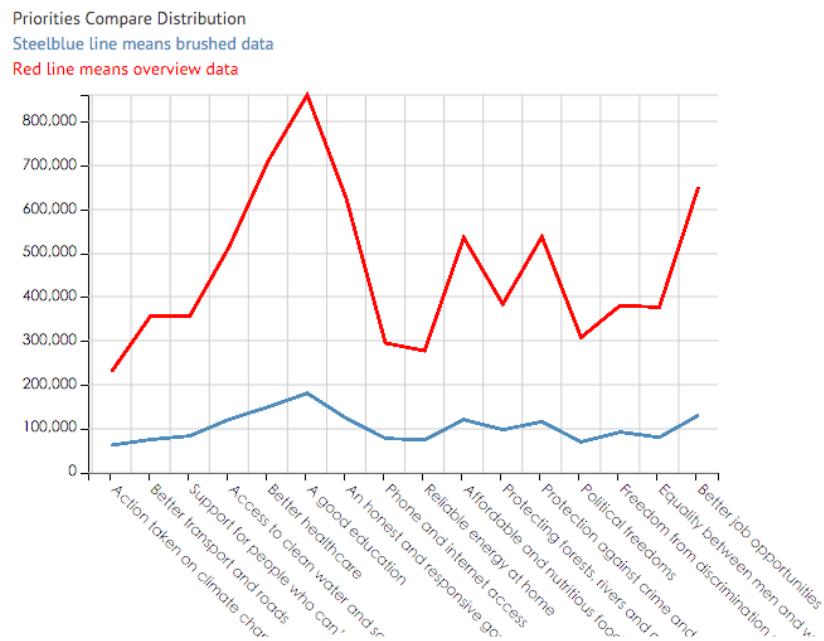
To solve the comparative issue of priority chart, there are two solutions.

1) Use comparative bar chart



The overview data are represented by the red rectangles while the selected data are represented by the blue rectangles. In this way, we can compare the overview data and the data selected by the brush by the height of the rectangles.

2) Use line chart



The red line represents the overview data while the blue line represents selected data. In this way, we can compare the overview data and the data selected by the brush by the height of the lines.

2. Task 4c

The screenshot of the source code:

```
104      // Store the original display data
105      self.origData = [];
106 ▼    self.displayData.forEach(function(d, i){
107        self.origData[i] = self.displayData[i];
108      });
109
110      //draw the verticle grid
111
112      var xGrid = self.visG.selectAll("lineX")
113        .data(self.origData);
114      xGrid.enter()
115        .append("line")
116        .attr("x1", function(d, i){
117          return self.xScale(i);})
118        .attr("x2", function(d, i){
119          return self.xScale(i);})
120        .attr("y1", 0)
121        .attr("y2", self.graphH)
122        .style("stroke", "#ccc");
123
124      //draw the horizontal grid
125      var yGrid = self.visG.selectAll("lineY")
126        .data(dataSet);
127      yGrid.enter()
128        .append("line")
129        .attr("x1", 0)
130        .attr("x2", self.graphW)
131 ▼    .attr("y1", function(d, i){
132      return d;
133    })
134 ▼    .attr("y2", function(d, i){
135      return d;
136    })
137    .style("stroke", "#ccc");
138
```

```

169
170 //Line Chart
171 //Brushed data|
172
173 var line = d3.svg.line()
174 .x(function(d,i) {
175     return self.xScale(i) + self.graphW/32; })
176 .y(function(d,i) {
177     return self.yScale(d); })
178 .interpolate("linear");
179
180 var path = self.visG.selectAll(".path").data(self.displayData);
181 path.exit().remove();
182 path.enter()
183 .append("path")
184 .attr({
185     "class": "path"
186 })
187 .style({
188     "stroke": "steelblue",
189     "stroke-width": 2,
190     "fill": null,
191     "fill-opacity":0
192 });
193 path.attr("d", line(self.displayData));
194
195 //Origdata
196 var line = d3.svg.line()
197 .x(function(d,i) {
198     return self.xScale(i) + self.graphW/32; })
199 .y(function(d,i) {
200     return self.yScale(d); })
201 .interpolate("linear");
202
203 var path = self.visG.selectAll(".OrigPath").data(self.origData);
204
205 path.enter()
206 .append("path")
207 .attr({
208     "class": "OrigPath"
209 })
210 .style({
211     "stroke": "red",
212     "stroke-width": 2,
213     "fill": null,
214     "fill-opacity":0
215 });
216 path.attr("d", line(self.origData));
217

```

I used two lines to implement the comparison of the overview data and the selected data. In order to draw two different lines, I stored the original display data (the y axis value) into another variable, so that each line would load their own data. To represent the data clearly, I have added the x scale and y scale grids.