EECS 6893 HW4

Jing Qian (jq2282)

Problem 1

1. Answer these questions in simple words.

1.1

SVG Coordinate Space and Mathematical/Graph Coordinate are both 2-dimensional flat space. They work in the same way except following:

- The (0, 0) coordinates of Mathematical Coordinate Space fall on the bottom left while those of SVG Coordinate Space on the top left.
- The Mathematical Coordinate Space has Y coordinate growing from bottom to top while SVG Coordinate Space has Y coordinate growing from top to bottom.

1.2

In d3.js, enter() and exit() are operations that used in joining an array of data to a D3 selection. If the array is longer than the selection, we use enter() to represent the missing elements. If the array is shorter than the selection, we use exit() to represent elements that need to be removed.

1.3

In SVG, transform functions are used for geometric manipulations on the SVG elements, including translation (movement), rotation, scale, and skew (shear).

A translation moves all the points of an element in the same direction and by the same amount. Translation preserves parallelism, angles and distances.

1.4

The function without a name is called an "anonymous function". In d3, we use them primarily to gain access to bound data. The anonymous function a.map(function(d,i){return i+5}) here equals to the following command in Python: [i+5 for i, d in enumerate(a)]. So the return value of the anonymous function is: [5, 6, 7, 8, 9].

2. Modify the sample code to get desired figure.

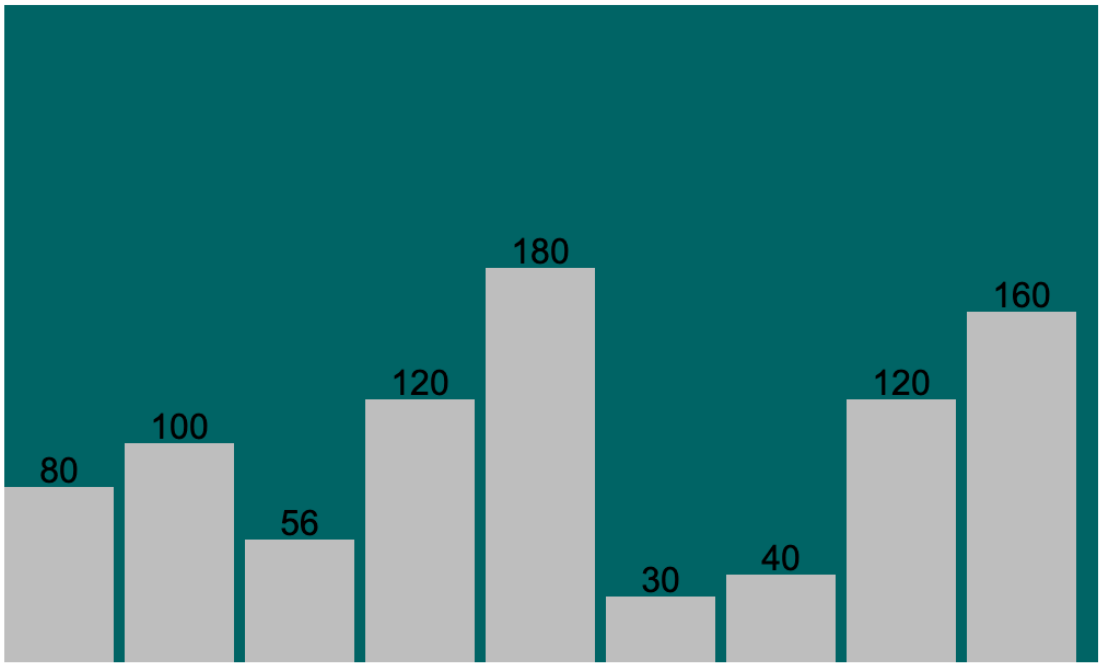
The screenshots of codes and the output bar-chart are as following:

big_data_hw4_part1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script type="text/javascript" src='HW4_Q1.js'> </script>
    <meta charset="UTF-8">
    <title>Homework 4 Question 1</title>
</head>

<style>body{}.bar-chart{background-color: #006666;}</style>
<svg class="bar-chart"></svg>

<script src="https://d3js.org/d3.v5.min.js"></script>
<body>
<script>
    HW4_Q1();
</script>

</body>
</html>
```

HW4_Q1.js

```
function HW4_Q1(){
    var data = [80, 100, 56, 120, 180, 30, 40, 120, 160];
    var svgWidth = 500, svgHeight = 300;
    // The required padding between bars is 5px.
    // The label must locate 2px above the middle of each bar.

    var svg = d3.select('svg')
            .attr("width", svgWidth)
            .attr("height", svgHeight);

    var barChart = svg.selectAll("rect")
            .data(data)
            .enter()
            .append("rect")
            .attr("class", "bar")
            .attr("transform", function(d, i) { return "translate(" + i * 55 + ","+
                (svgHeight-d)+")"; })
            .attr('width', 50)
            .attr('height', svgHeight)
            .attr("fill", "#C0C0C0");

    var text = svg.selectAll(".textlabel")
            .data(data)
            .enter()
            .append("text")
            .attr("class", "textlabel")
            .attr("transform", function(d, i) { return "translate(" + (i * 55+25) +
                ","+(svgHeight-d-2)+")"; })
            .attr("text-anchor", "middle")
            .style("font-family", "Arial")
            .style("color", "#333333")
            .text(function(d){return d;});
}
```



Problem 2

Step 1: Data processing

```
import numpy as np
import pandas as pd

data = pd.read_csv('wc.csv')
```

```
time = list(set(data.time))
df = pd.DataFrame(time, columns=['time'])
```

```
words = ['ai','data','good','movie','spark']
for i in words:
    df[i] = 0
```

```
for i in range(len(df)):
    for j in words:
        tmp = data[(data.time==df.time[i])&(data.word==j)]
        if not tmp.empty:
            df[j][i] = int(tmp['count'])
```

```
df.to_csv('wc2.csv', index=False, header=True)
```

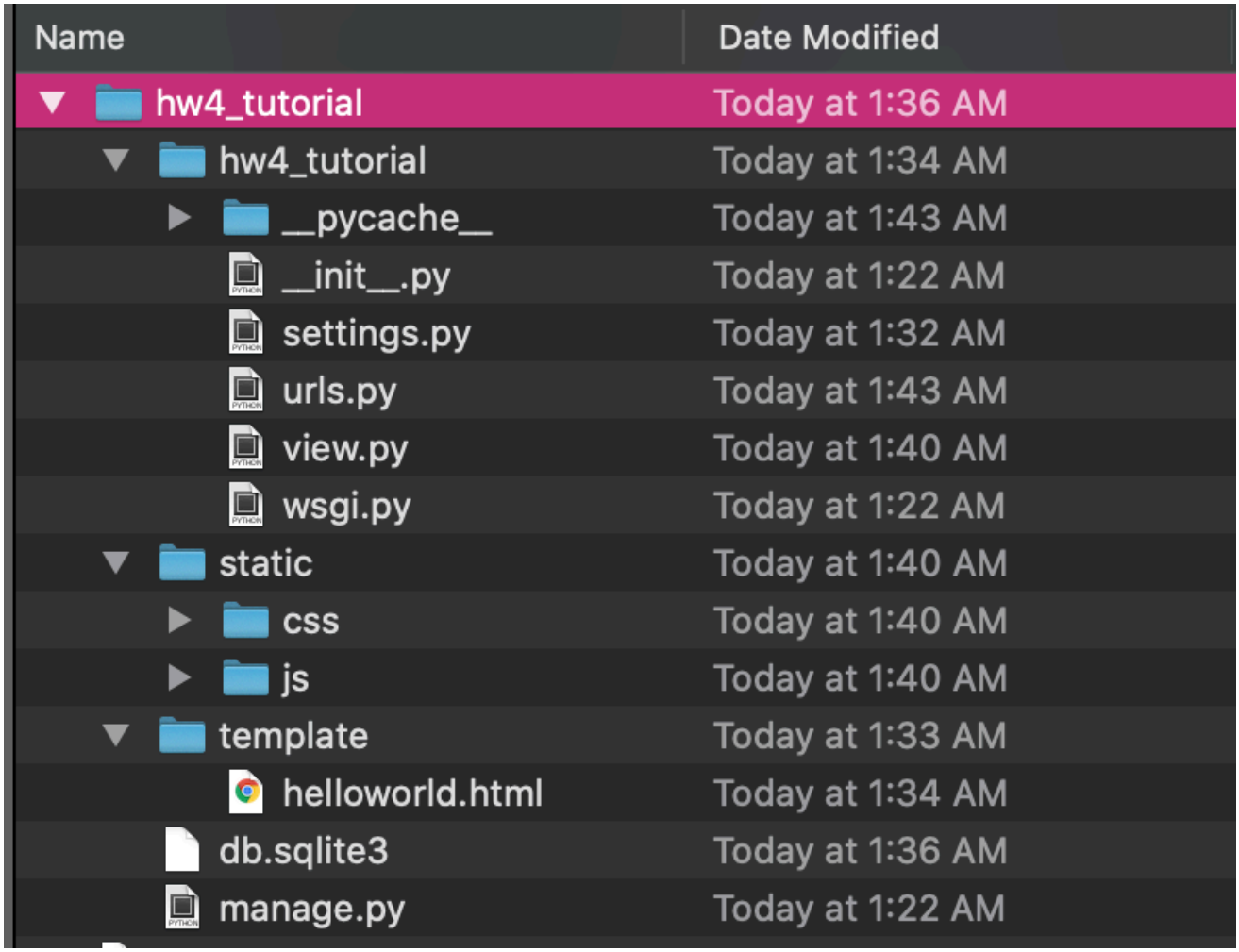
wc2

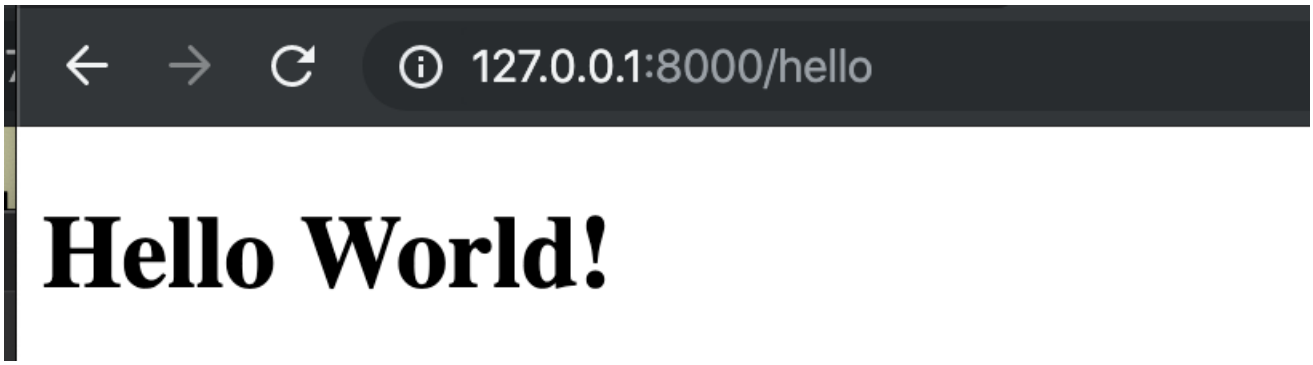
QUERY TABLE COPY

Schema Details Preview

Row	time	ai	data	good	movie	spark
1	2019-11-17 15:43:55 UTC	7	0	6	234	5
2	2019-11-17 15:48:55 UTC	11	2	6	191	13
3	2019-11-17 15:55:55 UTC	8	2	8	191	9
4	2019-11-17 15:42:55 UTC	12	2	8	185	5
5	2019-11-17 15:46:55 UTC	5	3	4	224	8
6	2019-11-17 15:47:55 UTC	12	3	10	209	9
7	2019-11-17 15:54:55 UTC	23	4	12	188	10

Step 2: Create Django project





Step 3. Finish the code

Modify the view.py:

```
    SQL = "select * from datasetHW3.wc2 limit 8"
    df = pandas_gbq.read_gbq(SQL)
    #print(df)
    data = {}

    res = []
    words = ['ai','data','good','movie','spark']
    time = sorted(list(df.time))
    for i in time:
        tmp = {}
        tmp['Time'] = str(i)[11:16]
        tmp['count'] = {}
        for j in words:
            tmp['count'][j] = int(df[df.time==i][j])
        res.append(tmp)
    data['data'] = res
    #print(data['data'])
    return render(request, 'dashboard.html', data)
```

Modify the dashboard.js

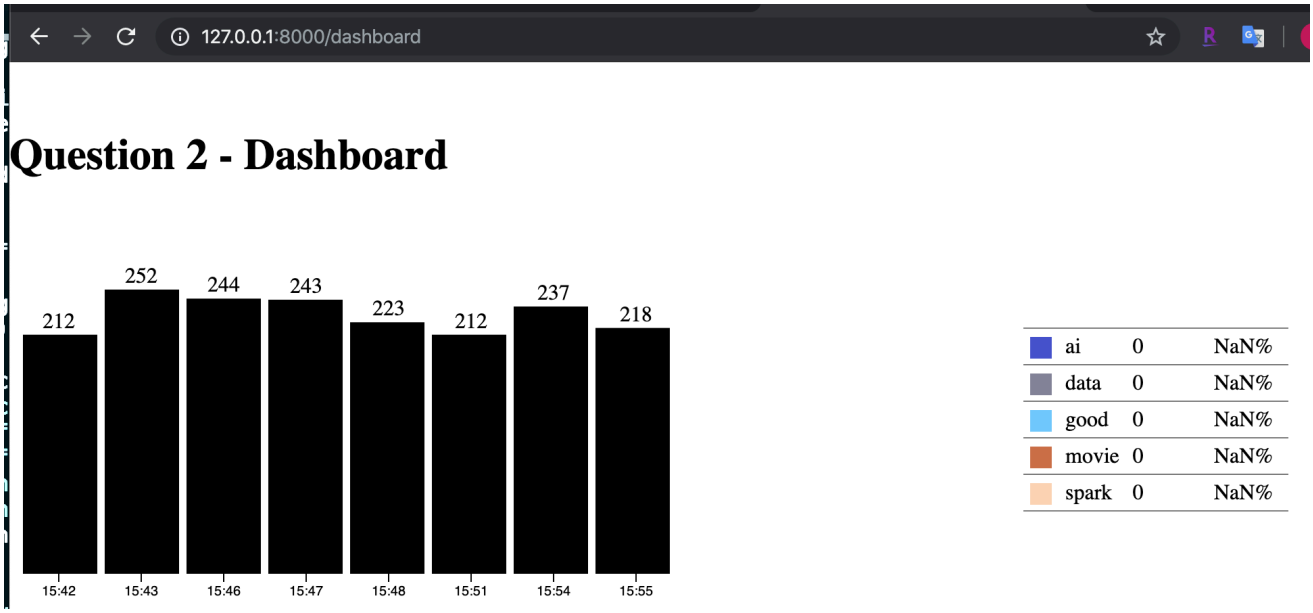
```
function segColor(c) {
    cmap = {ai: "#4753CC", data: "#828499", good: "#73C9FF", movie: "#CC6E47"
        , spark: '#FFD4B3'};
    return cmap[c];/* TO FINISH */
}

// compute total for each state.
fData.forEach(function (d) {
    d.total = d['count']['ai']+d['count']['data']+d['count']['good']+d['count'
        ]['movie']+d['count']['spark']; /* TO FINISH */
})
```

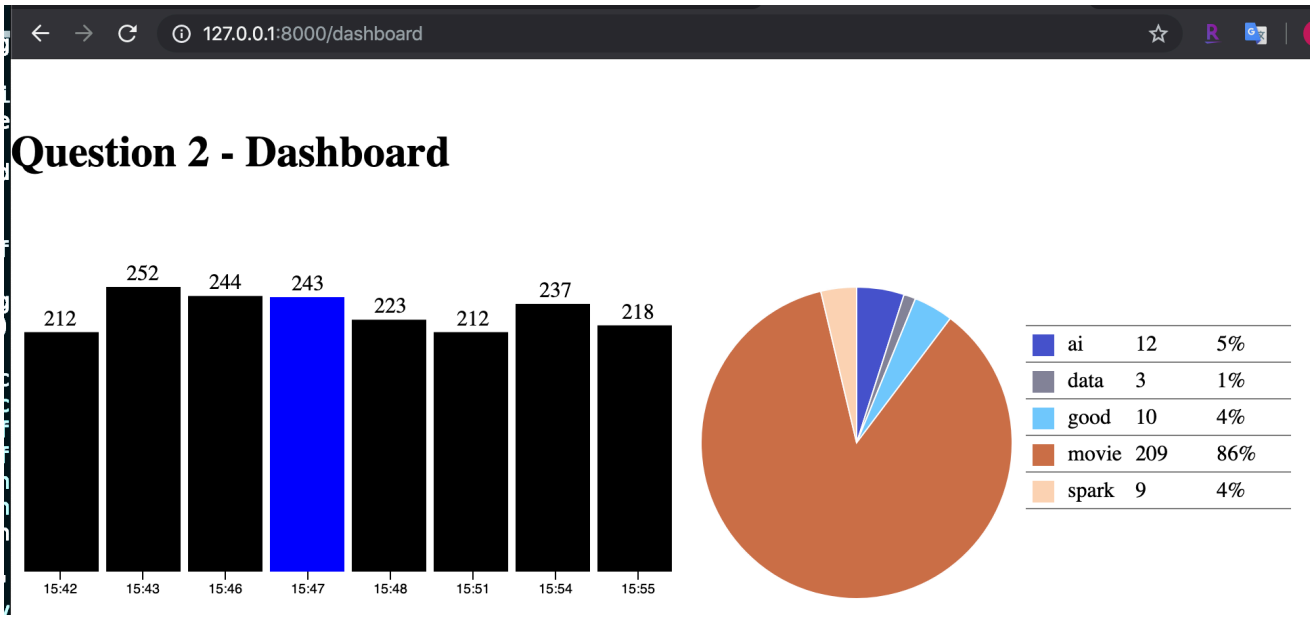
```
//create the rectangles.
bars.append("rect")
    .attr("x", function (d){return x(d[0])}/* TO FINISH */)
    .attr("y", function (d){return y(d[1])}/* TO FINISH */)
    .attr("width", x.rangeBand())
    .attr("height", function (d) {
        return hGDim.h - y(d[1]);
    })
    .attr('fill', barColor)
    .on("mouseover", mouseover)
    .on("mouseout", mouseout);

//Create the frequency labels ABOVE the rectangles.
bars.append("text").text(function (d) {
    return d3.format(",")(d[1])
})
    .attr("x", function (d) {
        return x(d[0])/* TO FINISH */ + x.rangeBand() / 2;
    })
    .attr("y", y(d[1])-5/* TO FINISH */)
    .attr("text-anchor", "middle");
```

```
// transition the height and color of rectangles.
bars.select("rect").transition().duration(500)
    .attr("y", function (d) {
        return y(d[1]);
    }/* TO FINISH */)
    .attr("height", function (d) {
        return hGDim.h - y(d[1]);
    })
    .attr("fill", color);
```

```
// calculate total count by segment for all state.
var tF = ['ai', 'data', 'good', 'movie', 'spark'].map(function (d) {
    return {
        type: d, count: d3.sum(fData.map(function (t) {
            return d.total/* TO FINISH */;
        }))
    };
});
```

Result:





Problem 3

Based on the edge.csv and node.csv provided by TA, the view.py is modified as following:

```
SQL1 = 'select * from datasetHW3.node'
df1 = pandas_gbq.read_gbq(SQL1)
#print("node preview: ", df1[:5])
node = []
for i in range(len(df1)):
    tmp = {}
    tmp['node'] = int(df1['node'][i])
    node.append(tmp)
#print("node,", node[:5])

SQL2 = 'select * from datasetHW3.edge'
df2 = pandas_gbq.read_gbq(SQL2)
#print("edge preview: ", df2[:5])
edge = []
visited = set()
for i in range(len(df2)):
    tmp = {}
    pair = (int(df2['source'][i]), int(df2['target'][i]))
    if pair not in visited:
        visited.add(pair)
        tmp['source'] = pair[0]
        tmp['target'] = pair[1]
        edge.append(tmp)
print("len of edge: ", len(edge))
data = {}
data['n'] = node
data['e'] = edge
```

Modified connection.js:

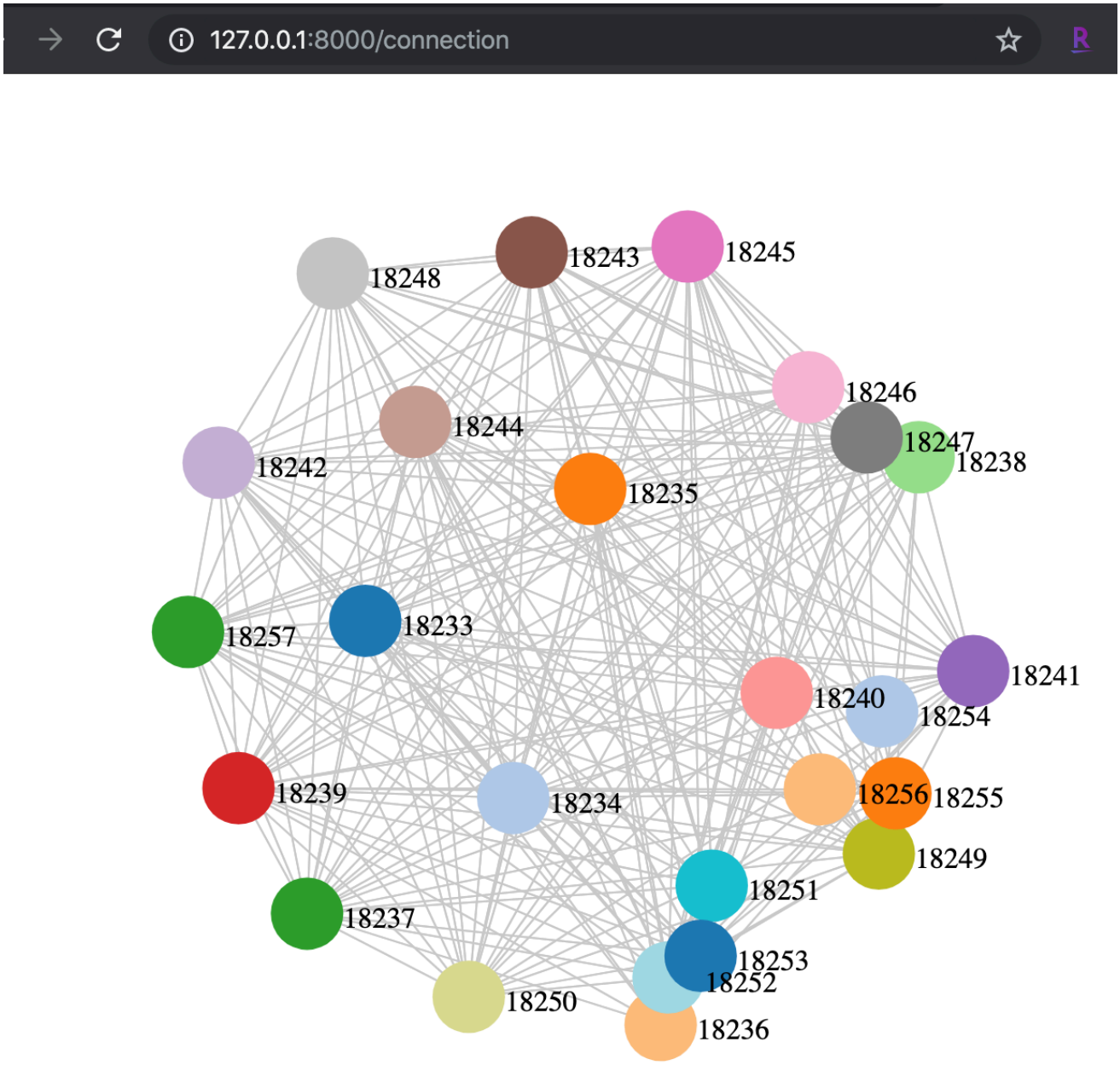
```
var svg = d3.select("body")
            .append("svg")
            .attr("width", width/* TO FINISH */)
            .attr("height", height/* TO FINISH */);
```

```
var svg_edges = svg.selectAll("line")
                   .data(edges/* TO FINISH */)
                   .enter()
                   .append("line"/* TO FINISH */)
                   .style("stroke","#ccc")
                   .style("stroke-width",1);
```

```
var svg_nodes = svg.selectAll("circle")
                   .data(nodes/* TO FINISH */)
                   .enter()
                   .append("circle"/* TO FINISH */)
                   .attr("r",20)
                   .style("fill", function(d,i){return color(i);}/* TO FINISH
                   */)
                   .call(force.drag);
```

```
var svg_texts = svg.selectAll("text")
                   .data(nodes)
                   .enter()
                   .append("text")
                   .style("fill", "black")
                   .attr("dx", 20)
                   .attr("dy", 8)
                   .text(function(d){return d.node;}/* TO FINISH */ );


force.on("tick", function(){
     svg_edges.attr("x1", function(d){return d.source.x;}/* TO FINISH */)
              .attr("y1", function(d){return d.source.y;}/* TO FINISH */)
              .attr("x2", function(d){return d.target.x;}/* TO FINISH */)
              .attr("y2", function(d){return d.target.y;}/* TO FINISH */);
```

Result:


127.0.0.1:8000/connection
18243
18245
18248
18246
18244
18247
18238
18242
18235
18257
18233
18241
18240
18254
18239
18234
18256
18255
18249
18251
18237
18253
18252
18250
18236