3803ICT

Big Data Analysis

Lab 05 – Data Visualization and Visual Analysis

Trimester 1 - 2020

Table of Contents

I. Tutorials

1. D3JS Introduction

D3.js (or just D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, **interactive** data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards. (https://en.wikipedia.org/wiki/D3.js)

More details on syntax and how to use D3.js can be found on https://d3js.org/ and https://square.github.io/intro-to-d3/.

A straight-forward way to use D3.js out of the box is browsing its gallery at https://github.com/d3/d3/wiki/Gallery. We choose a suitable data visualization and convert our data to its format.

2. FIFA World Cup Dataset

The current visualization of FIFA World Cup dataset (http://www.fifa.com/fifa-tournaments/archive/worldcup/index.html) does not have a single unified view. We want to implement an interactive and unified data visualization. You are provided a statistics file "team_statistics.json" with a couple of other auxiliary files.

3. Running the web page

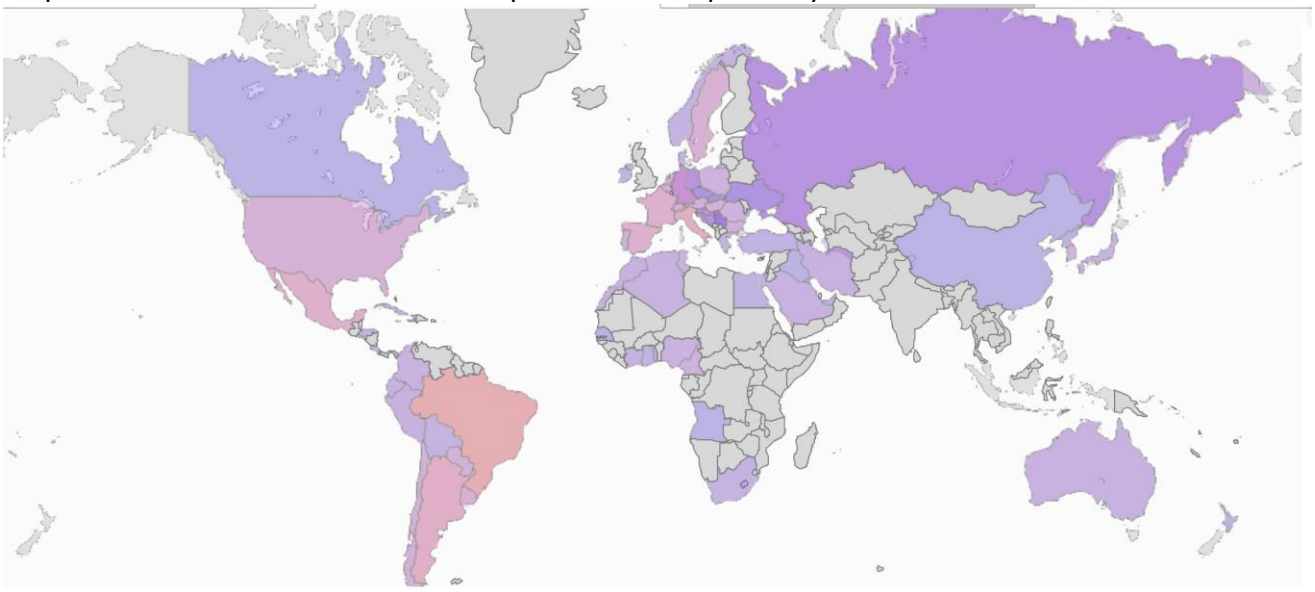
Step 1. Go to the folder "worldcupviz" and start a web server via the terminal:

python -m SimpleHTTPServer (Python 2)
python3 -m http.server (Python 3)

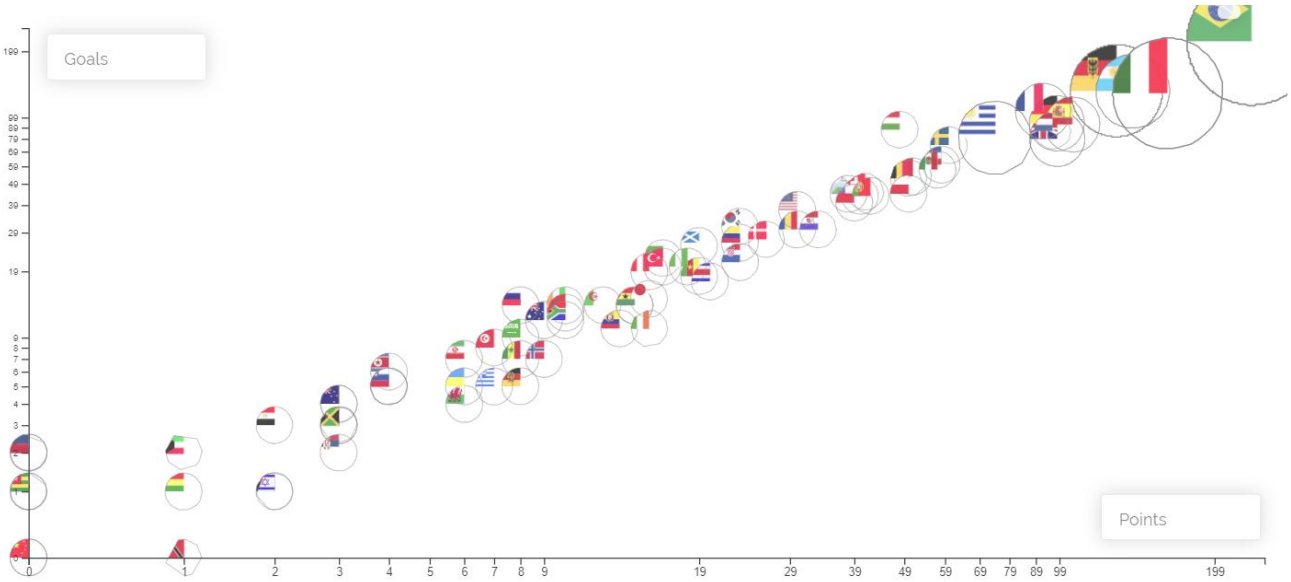
Step 2. Read the address in the output of the above command and type it in a web browser:

http://0.0.0.0:8000/

Step 3. You will see the Attendance map as follows. Explore it yourself.



Step 4. Go to the Bubble tab, you will see:

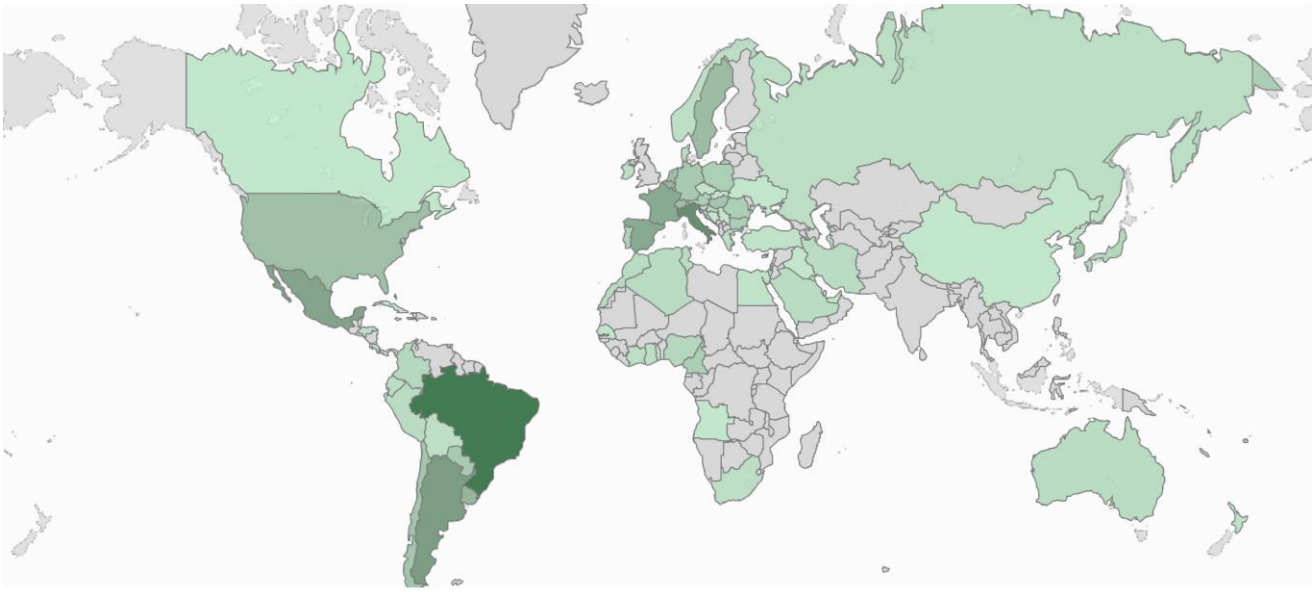


II. Practices

1. Change the colour of choropleth map

Investigate the Javascript file “homePage.js” in the folder “worldcupviz”.

- Find the code that allows to change the colour of the choropleth map
- Find a way to change the colour of the above map to:



HINT: explore a color map at https://color.adobe.com/ and https://github.com/d3/d3-interpolate

2. Implement the slider

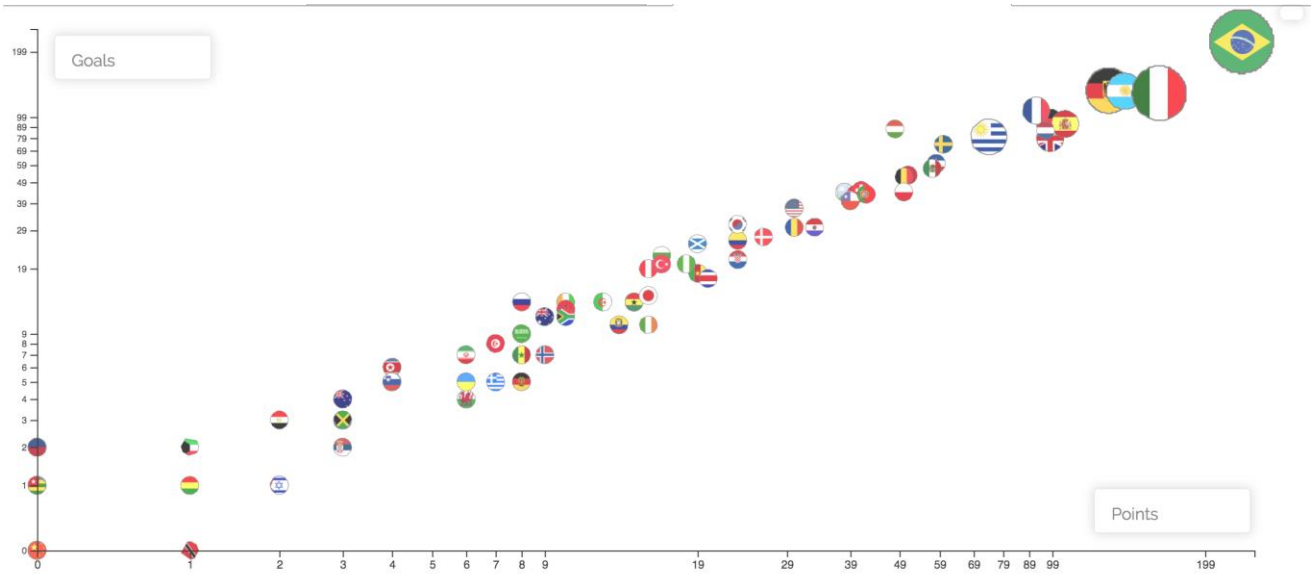
We already put the Slider bar for you:

Fill in the TODO code in the Homepage.js file to have an interactive visualization (i.e. data will be limited from the oldest world cup to the chosen year):

```
// TODO
// currentYear = ;
// output.innerHTML = ;
```

3. Improve the circle radius

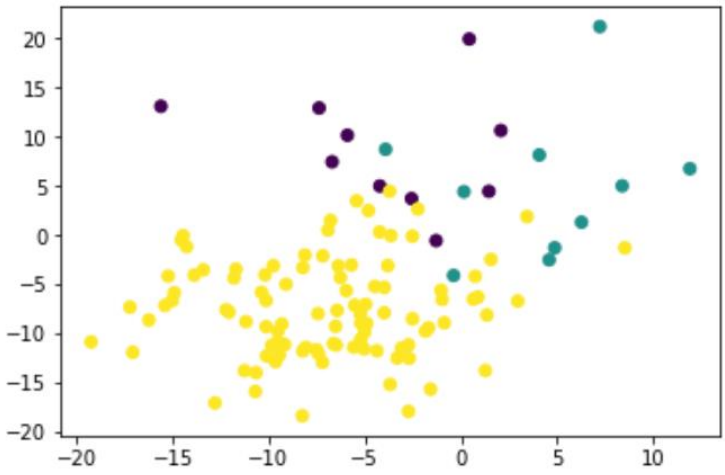
- Find the code that allows to change the radius of the circle markers in the Bubble tab
- Change the radius of the circles to a reasonable value as shown below:



III. Handling the Volume of Big Data

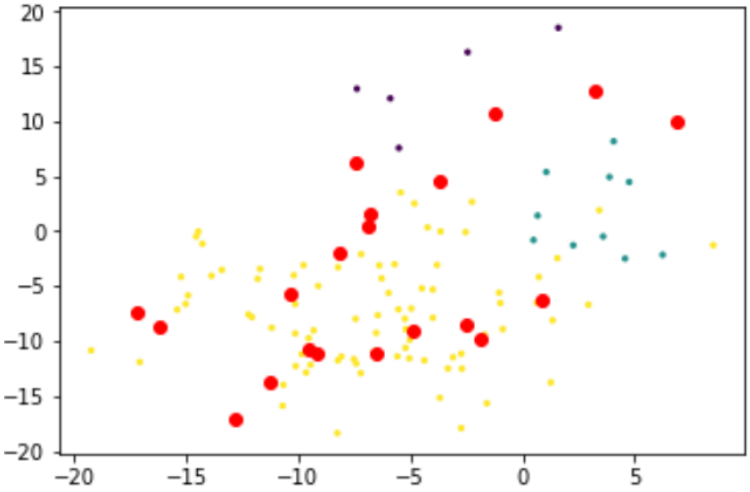
In this section, you will explore how to handle the volume of big data by sampling. Firstly, we will load the synthetic 2-D data points by scikit-learn.

Now you will visualize the dataset. Complete the code marked with TODO (in the attached Jupyter notebook).



1. Simple Random Sampling (SRS)

Now we implement the first sampling method, called Simple Random Smapling (SRS). In this method, there is an equal probability of selecting any item. Complete the code marked with TODO in the attached Jupyter notebook).

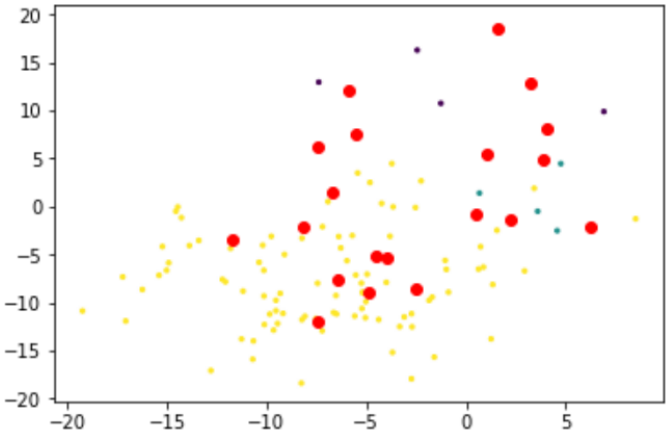


2. Stratified Sampling

Now we will explore another sampling method, called stratified sampling. This method comprises of two steps:

1. Split the data into several partitions. We will use k-mean clustering for this purpose
2. Draw random samples from each partition. For simplicity, we make the stratified random samples of the same size.

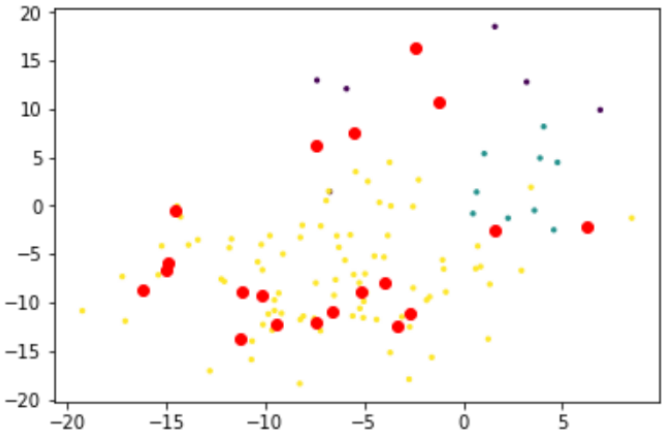
Complete the code marked with TODO in the attached Jupyter notebook).



Now we want to improve stratified sampling such that:

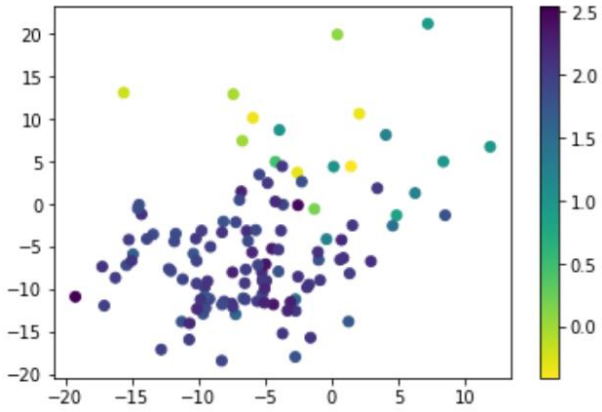
- The stratified random samples are not of the same size but proportion to the size of clusters
- Each cluster has at least one sampling points

Complete the code marked with TODO (in the attached Jupyter notebook).



IV. Handling the Variety of Big Data (OPTIONAL)

In this section, we are going to implement a diversification algorithm, namely Motley, in the lecture. We reuse the dataset in the previous section. First, we are going to define the relevance of each data item by using the data label in ground truth and generating a random number around the mean equal to label value. Note that the color bar indicates the relevance degree.



1. Similarity function

We will implement the dissimilarity function by using Euclidean distance. Complete the code marked with TODO (in the attached Jupyter notebook).

```
array([[  0.        ,   5.44066286,  15.66888123, ...,  15.87679126,
         17.24853177,  17.14955479],
       [  5.44066286,   0.        ,  11.66141536, ...,  19.12350055,
         22.07171749,  21.73430097],
       [ 15.66888123,  11.66141536,   0.        , ...,  30.76298182,
         32.89912048,  32.81407466],
       ...,
       [ 15.87679126,  19.12350055,  30.76298182, ...,   0.        ,
          8.51128548,   6.89821455],
       [ 17.24853177,  22.07171749,  32.89912048, ...,   8.51128548,
          0.        ,   1.71166094],
       [ 17.14955479,  21.73430097,  32.81407466, ...,   6.89821455,
          1.71166094,   0.        ]])
```

2. Motley algorithm

The Motley algorithm constructs the output by incrementally adding items in the decreasing order of relevance and maximizing the minimum dissimilarity.

- Traverse items in the decreasing order of relevance
- Add an item to output if the dissimilarity with other selected items is larger than a threshold delta

Complete the code marked with TODO (in the attached Jupyter notebook).