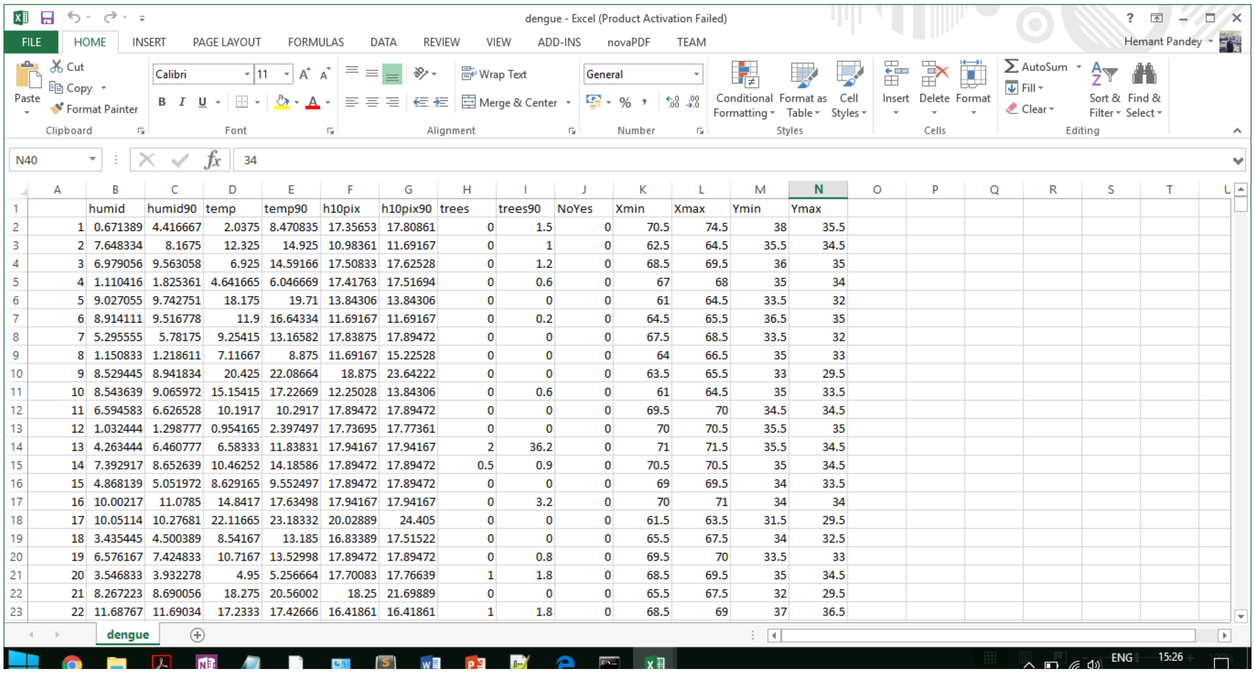
LAB 2

DATASET

Chosen Dataset: **Dengue prevalence, by administrative region**

Link to Dataset: https://vincentarelbundock.github.io/Rdatasets/csv/DAAG/dengue.csv

Dataset contains 2000 rows & 13 dimensions. All the attributes have numerical values.



	humid	humid90	temp	temp90	h10pix	h10pix90	trees	trees90	NoYes	Xmin	Xmax	Ymin	Ymax
1	0.671389	4.416667	2.0375	8.470835	17.35653	17.80861	0	1.5	0	70.5	74.5	38	35.5
2	7.648334	8.1675	12.325	14.925	10.98361	11.69167	0	1	0	62.5	64.5	35.5	34.5
3	6.979056	9.563058	6.925	14.59166	17.50833	17.62528	0	1.2	0	68.5	69.5	36	35
4	1.110416	1.825361	4.641665	6.046669	17.41763	17.51694	0	0.6	0	67	68	35	34
5	9.027055	9.742751	18.175	19.71	13.84306	13.84306	0	0	0	61	64.5	33.5	32
6	8.914111	9.516778	11.9	16.64334	11.69167	11.69167	0	0.2	0	64.5	65.5	36.5	35
7	5.295555	5.78175	9.25415	13.16582	17.83875	17.89472	0	0	0	67.5	68.5	33.5	32
8	1.150833	1.218611	7.11667	8.875	11.69167	15.22528	0	0	0	64	66.5	35	33
9	8.529445	8.941834	20.425	22.08664	18.875	23.64222	0	0	0	63.5	65.5	33	29.5
10	8.543639	9.065972	15.15415	17.22669	12.25028	13.84306	0	0.6	0	61	64.5	35	33.5
11	6.594583	6.626528	10.1917	10.2917	17.89472	17.89472	0	0	0	69.5	70	34.5	34.5
12	1.032444	1.298777	0.954165	2.397497	17.73695	17.77361	0	0	0	70	70.5	35.5	35
13	4.263444	6.460777	6.58333	11.83831	17.94167	17.94167	2	36.2	0	71	71.5	35.5	34.5
14	7.392917	8.652639	10.46252	14.18586	17.89472	17.89472	0.5	0.9	0	70.5	70.5	35	34.5
15	4.868139	5.051972	8.629165	9.552497	17.89472	17.89472	0	0	0	69	69.5	34	33.5
16	10.00217	11.0785	14.8417	17.63498	17.94167	17.94167	0	3.2	0	70	71	34	34
17	10.05114	10.27681	22.11665	23.18332	20.02889	24.405	0	0	0	61.5	63.5	31.5	29.5
18	3.435445	4.500389	8.54167	13.185	16.83389	17.51522	0	0	0	65.5	67.5	34	32.5
19	6.576167	7.424833	10.7167	13.52998	17.89472	17.89472	0	0.8	0	69.5	70	33.5	33
20	3.546833	3.932278	4.95	5.256664	17.70083	17.76639	1	1.8	0	68.5	69.5	35	34.5
21	8.267223	8.690056	18.275	20.56002	18.25	21.69889	0	0	0	65.5	67.5	32	29.5
22	11.68767	11.69034	17.2333	17.42666	16.41861	16.41861	1	1.8	0	68.5	69	37	36.5

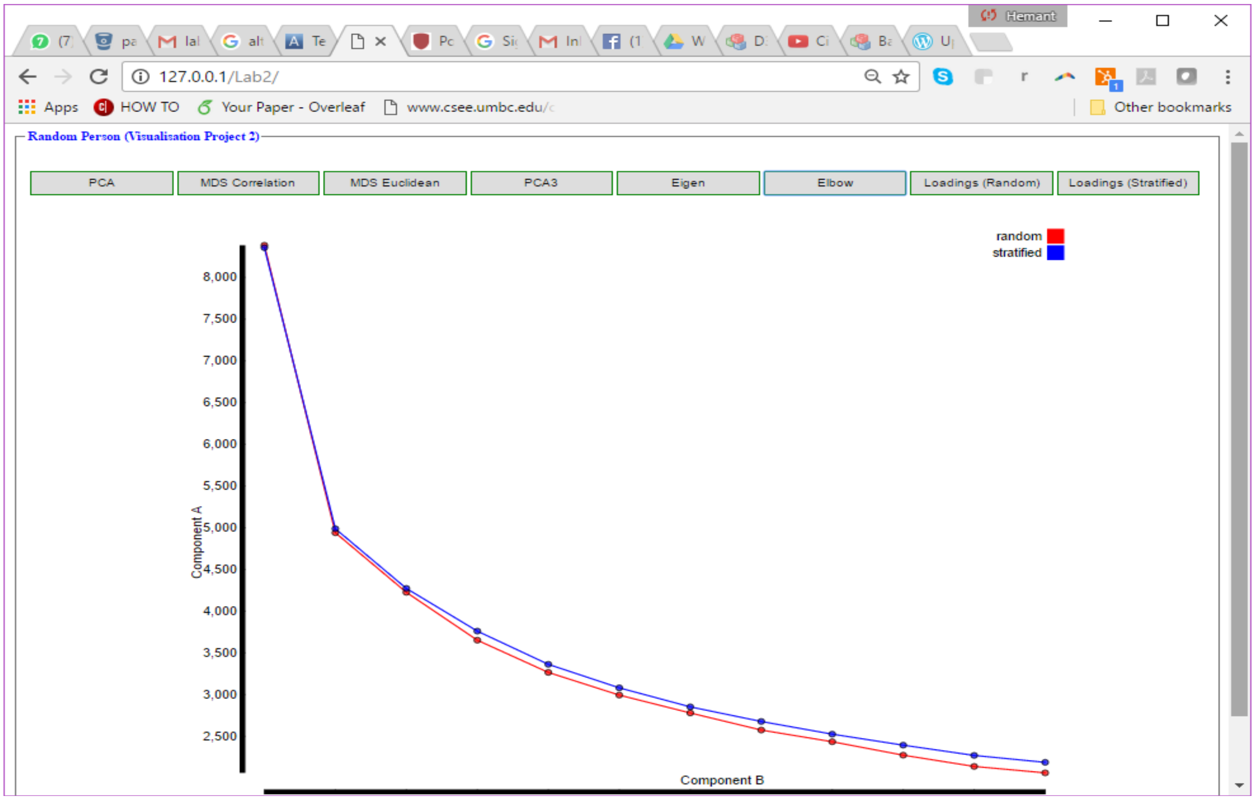
I have used client server system: python for processing (server), D3 (v3) for VIS (client).

TASKS AND IMPLEMENTATION

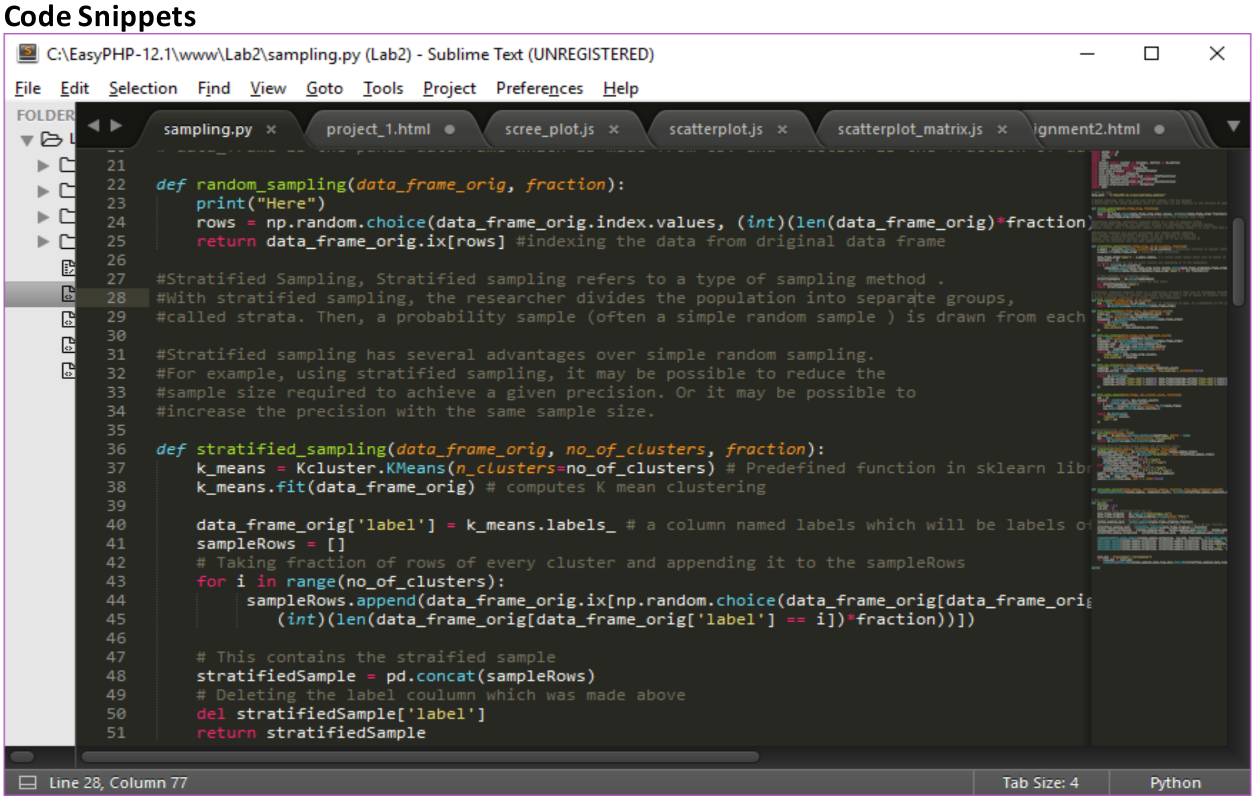
Task1: data clustering and decimation (30 points)

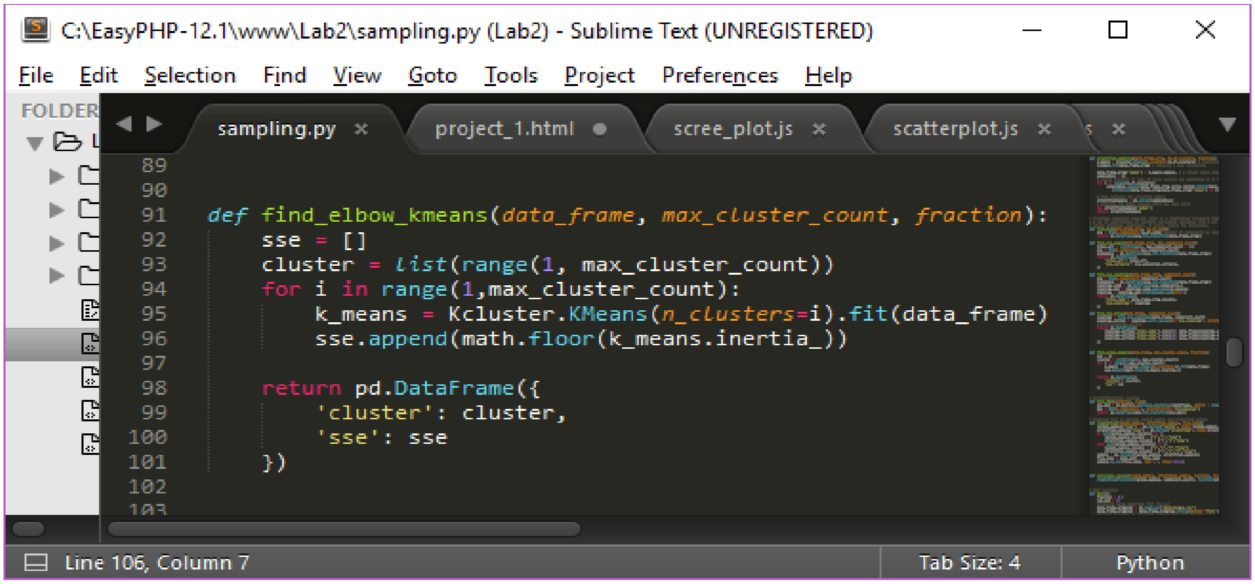
I used Python's numpy's random choice function to randomly select 30% for random sampling. For Stratified Sampling, I used K Means Clustering & constructed an elbow function to obtain the appropriate number of clusters, which turned out to be 4 and then selected 30% from each of the four clusters.

K means Elbow Plot



Code Snippets

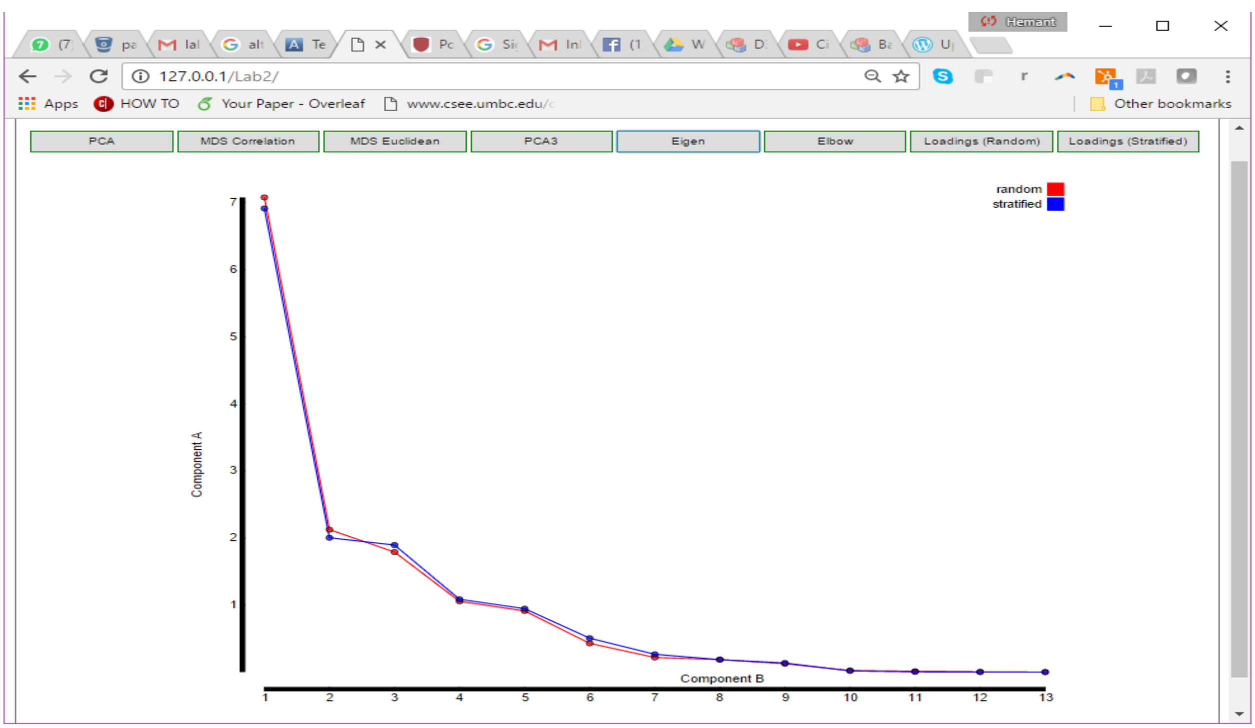




Task 2: dimension reduction (use decimated data) (30 points)

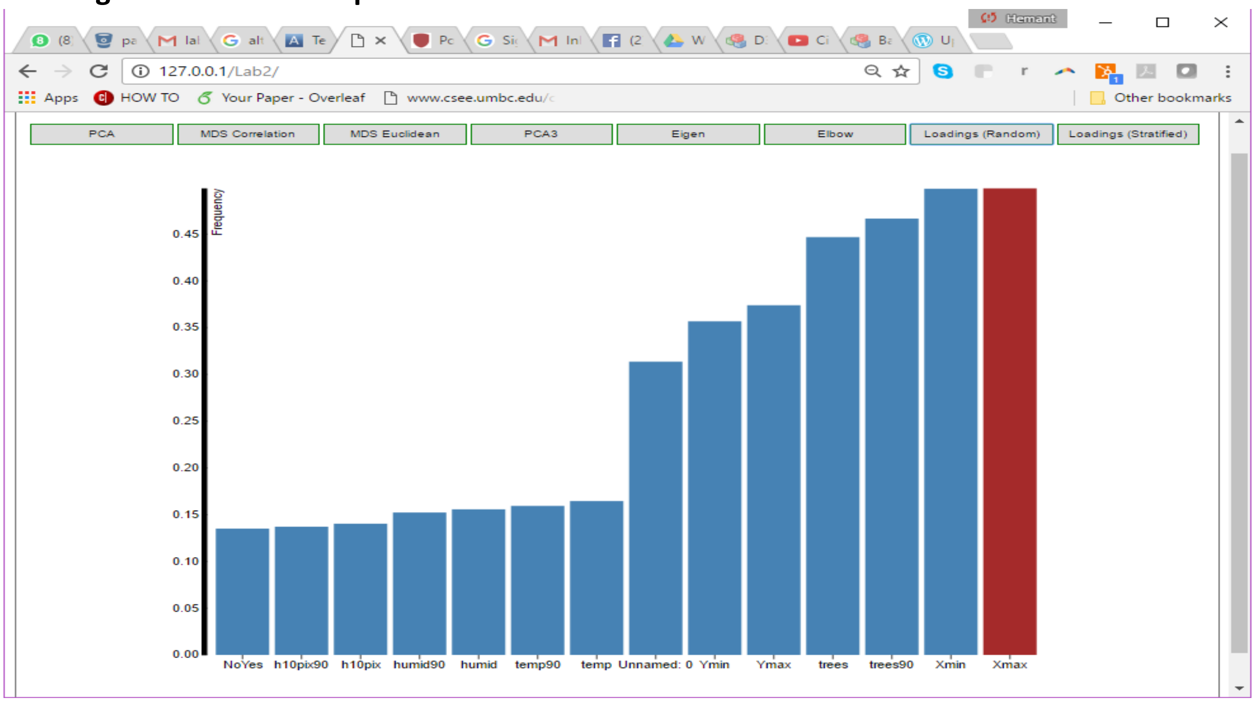
Scree plots for both the random samples and stratified samples were obtained as shown below. The intrinsic dimensionality of the data is all points obtained where the Eigen Values were > 1 i.e. 4.

Scree Plots for Random and Stratified Samples

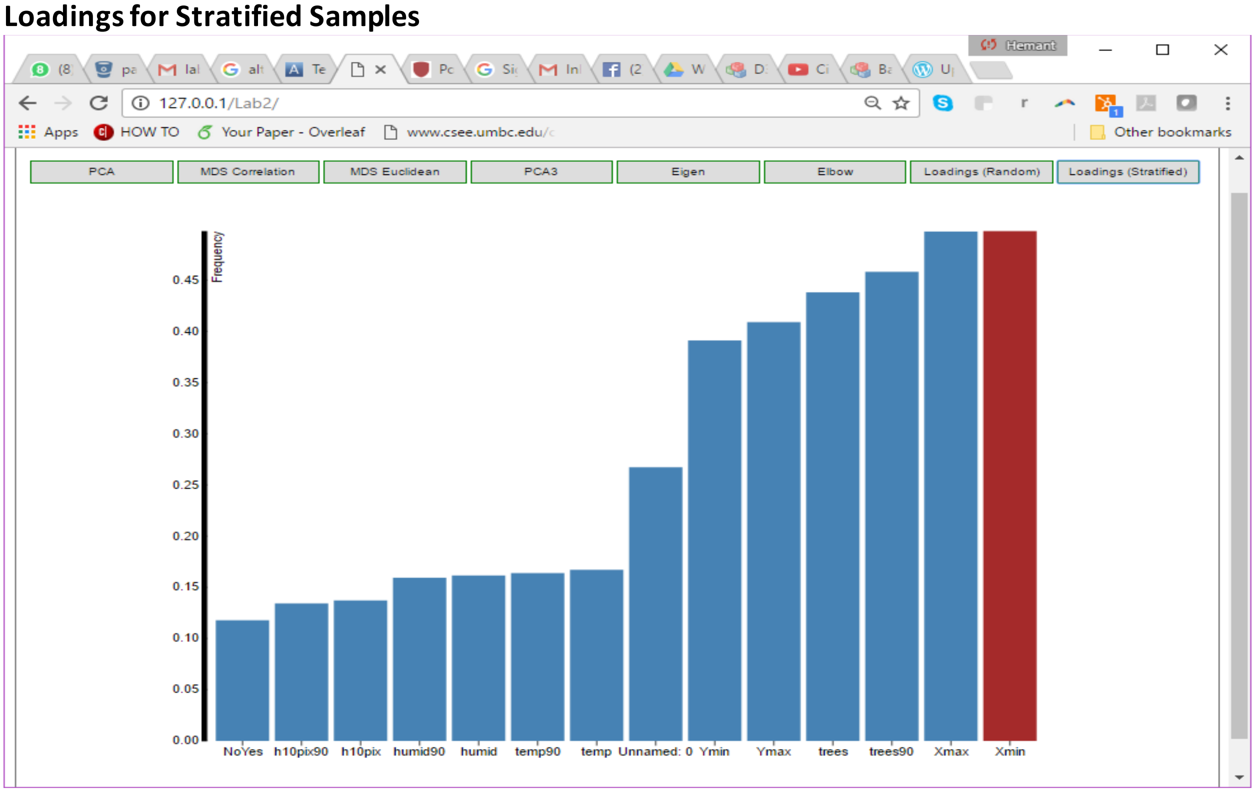


Now, we obtain the three highest attributes with highest PCA loadings and saved them for producing a scatter plot matrix later.

Loadings for Random Samples



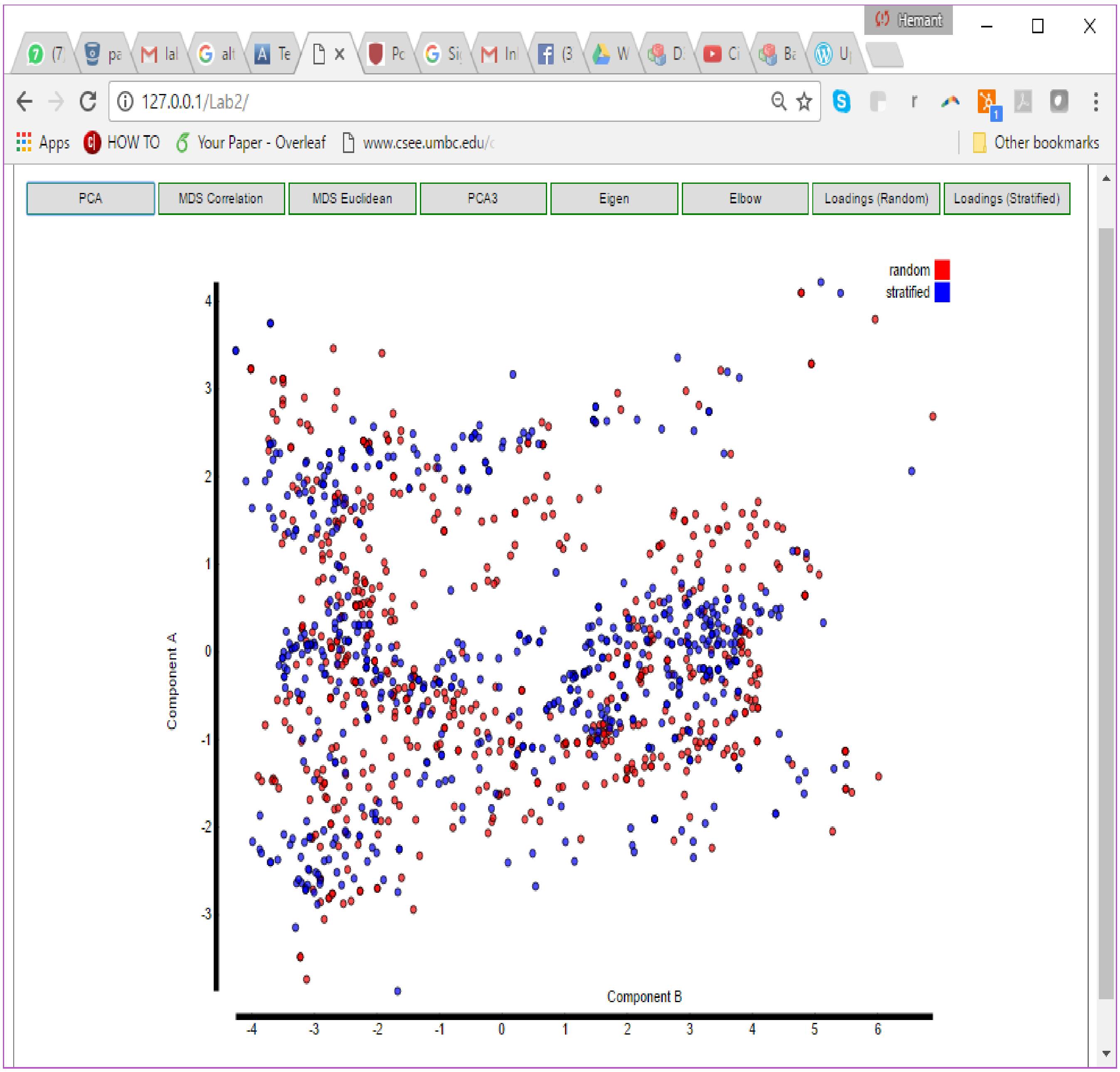
Loadings for Stratified Samples



To obtain the three highest attributes with maximum PCA loading, data is first reduced to only the intrinsic dimensions and then squared the loading value for all the attributes. Attributes with the highest sum of this squared value were considered as the three attributes with maximum PCA loadings.

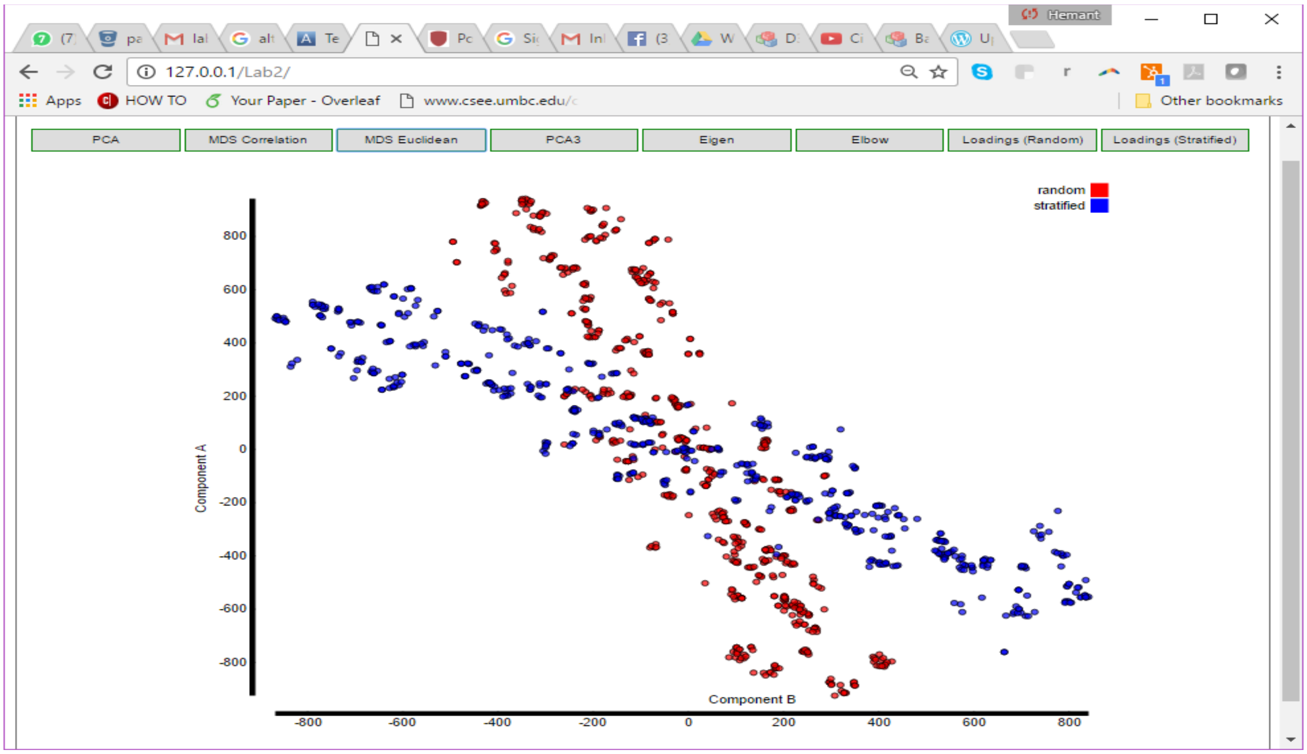
Task 3: visualization (use dimension reduced data) (40 points)

Visualize data projected into the top two PCA vectors via 2D scatterplot

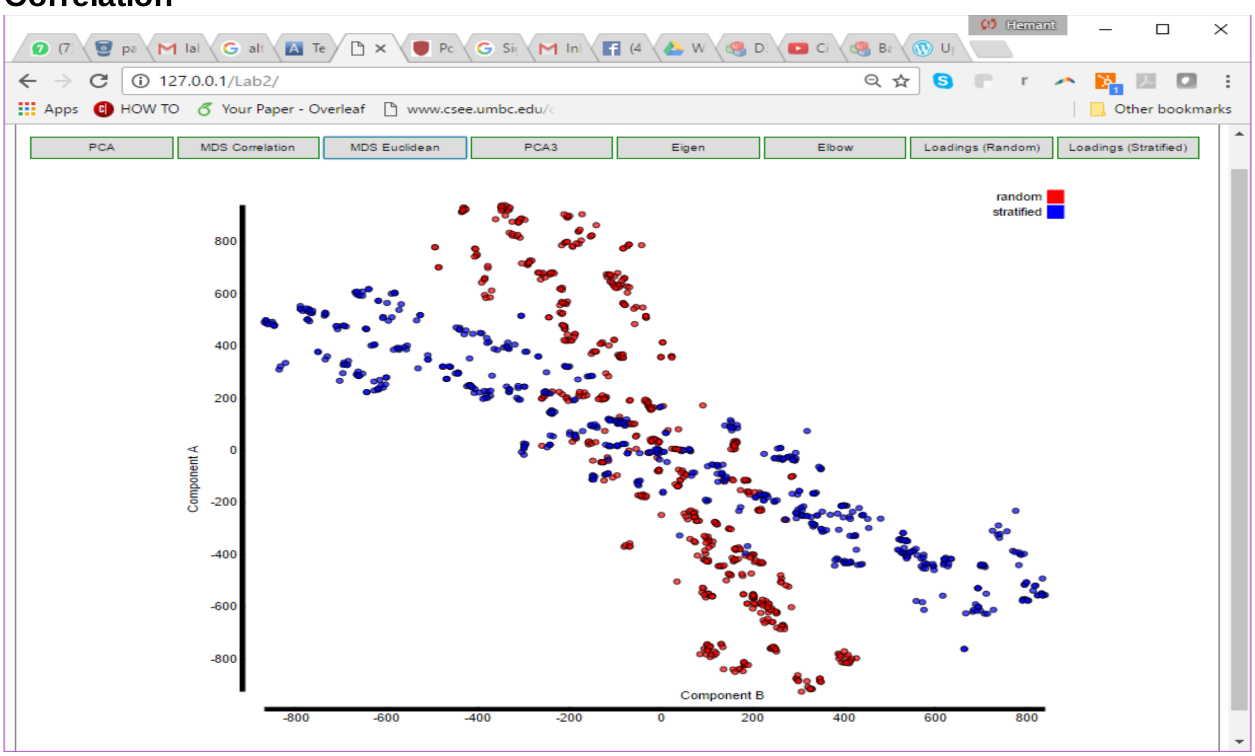


Visualize data via MDS (Euclidian & correlation distance) in 2D scatterplots

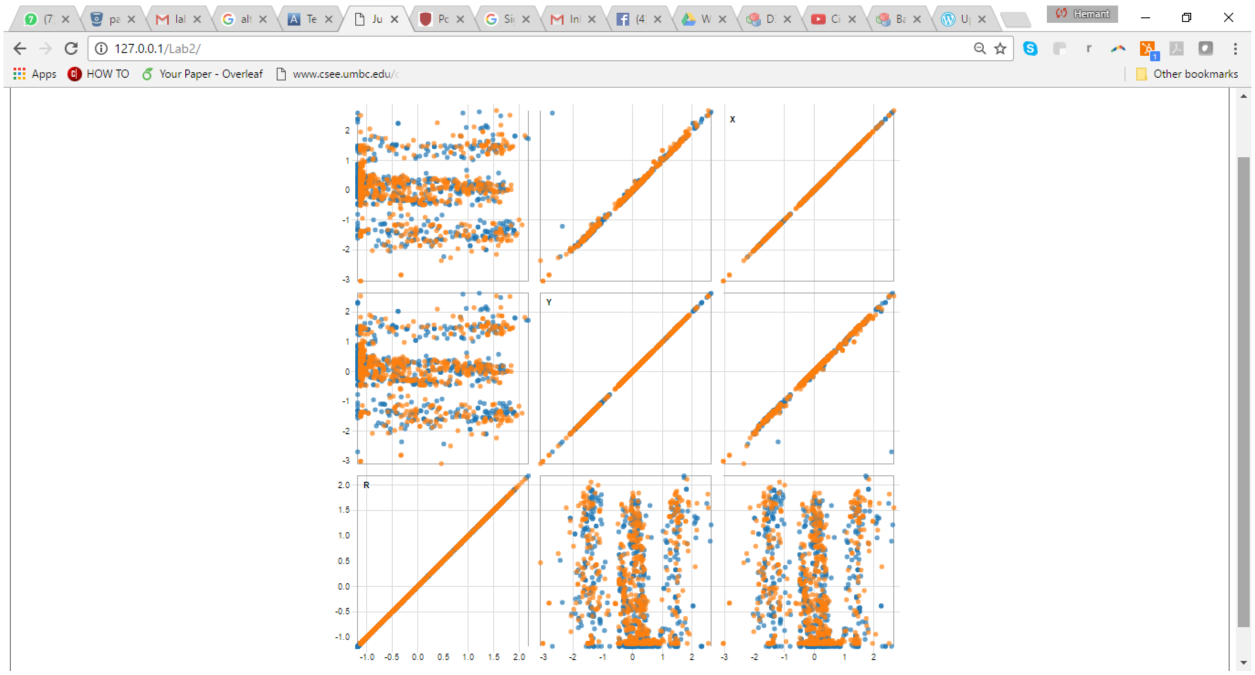
Euclidean



Correlation



Visualize scatterplot matrix of the three highest PCA loaded attributes



CODE SNIPPETS

Scatterplot.js

```
function plot_values(filename) {
    refresh();
    filename = "./data_sampled/" + filename;
    console.log(filename);
    svg.selectAll("*").remove();

    var color = ["Red","Blue"];
    //console.log(color[0]);
    var width = 940,
    size = 300,
    padding = 20;
    var left_pad = 100;
    // Load data
    var xScale = d3.scale.linear().range([left_pad, w-pad]);
    var xAxis = d3.svg.axis().scale(xScale).orient("bottom");

    var yScale = d3.scale.linear().range([h-pad*2,pad]);
    var yAxis = d3.svg.axis().scale(yScale).orient("left");

    d3.csv(filename, function(error, data) {
        data.forEach(function(d) {
            d.x = +d.x;
            d.y = +d.y;
            d.type = d.type
        });

        var xValueR = function(d) { return d.x;};
        var yValueR = function(d) { return d.y;};

       xScale.domain([d3.min(data, xValueR), d3.max(data, xValueR)]);
        yScale.domain([d3.min(data, yValueR), d3.max(data, yValueR)]);


        svg.append("g")
          .attr("class", "axis")
          .attr("transform", "translate(0, "+(h-pad)+")")
          .call(xAxis)
          .append("text")
        .attr("transform", "rotate(-90)")
        .attr("y", left_pad-80)
        .attr("x",h-400)
        .attr("dy", "0.7em")
        .style("text-anchor", "middle")
        .text("Component A")
```

```
        svg.selectAll("circle")
            .data(data)
            .enter()
            .append("circle")
            .attr("r", 3.5)
            .attr("cx", function(d){
                return xScale(d.x);
            })
            .attr("cy", function(d){
                return yScale(d.y);
            })
            .attr("fill", function(d) {

                return d.type === "random" ? color[0] : color[1];
            })
            .attr("stroke", "black")
            //.attr("stroke-width", function(d) {return d/2;});
            ;

        console.log("Circles printed for all samples")

        // draw legend
  var legend = svg.selectAll(".legend")
      .data(typeArr)
    .enter().append("g")
      .attr("class", "legend")
      .attr("transform", function(d, i) { return "translate(0," + i * 20 + ")"; });

  // draw legend colored rectangles
  legend.append("rect")
      .attr("x", width - 18)
      .attr("width", 18)
      .attr("height", 18)
      .style("fill", function(d, i) { return color[i]; });

  // draw legend text
  legend.append("text")
      .attr("x", width - 24)
      .attr("y", 9)
      .attr("dy", ".35em")
      .style("text-anchor", "end")
      .text(function(i) { return i;})
```

Scatterplot_matrix.js

```
function plot_matrix(filename) {
    refresh();

filename = "./data_sampled/" + filename;
console.log(filename);
var svg = d3.select("svg");
svg.selectAll("*").remove();

var width = 940,
    size = 300,
    padding = 20;

var x = d3.scale.linear()
    .range([padding / 2, size - padding / 2]);

var y = d3.scale.linear()
    .range([size - padding / 2, padding / 2]);

var xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom")
    .ticks(6);

var yAxis = d3.svg.axis()
    .scale(y)
    .orient("left")
    .ticks(6);

var color = d3.scale.category10();

d3.csv(filename, function(error, data) {
  if (error) throw error;

  data.forEach(function(d) {
    d.type = d.type
  });

  var domainByTrait = {},
      traits = d3.keys(data[0]).filter(function(d) { return d !== "type"; }),
      n = traits.length;
      console.log(traits);
```

The complete code and the video can be seen in the attachments of submission.

Interesting observations about data

The dataset had 13 attributes, out of which only 4 has eigen value 1 or higher. The intrinsic dimensionality of the data was 4 for a dataset of 13 attributes. The data of some attributes varies largely as compared to data of other attributes, thus it becomes essential to normalize them so that one attribute does not completely overwhelm the other.

Instead of using random and stratified sampling, reservoir or inversion sampling could also be used. Reservoir sampling is more appropriate when we have streaming data. Since we have a fixed dataset, stratified sampling seems to be the most appropriate sampling technique which also gives us a uniform sample data spread across all the clusters. Similarly, instead of using PCA, we can use techniques like LDA, GDA, Isomap etc.

REFERENCES

1.) Scatter Plot reference:
https://bl.ocks.org/mbostock/3887118

2.) Scatter Plot Matrix reference
https://bl.ocks.org/mbostock/3213173

3.) Bar Chart Reference
https://bl.ocks.org/mbostock/3885304