# Visualization Mini Project 2

1. Introduction:

   In this assignment, we learn to

   (1) Use Python to import CSV file and pre-process a large amount of data

   (2) Use Python to reduce data by random sampling and adaptive sampling with K-Means clustering

   (3) Use Python to perform dimension reduction by using Principal component analysis (PCA) and Multidimensional scaling (MDS)

   (4) Use the d3 package for elegant visual effect and animation for data visualization.

   (5) Build a user friendly interface via basic HTML and CSS techniques.

2. Requirement:

**Practice the three basic tasks of visual data analytics**

- use data from mini project #1 (or other), begin with |N|≥500, |D|≥10)
- client-server system: python for processing (server), D3 for VIS (client)

**Task1: data clustering and decimation (30 points)**

- implement random sampling and stratified sampling
- the latter includes the need for k-means clustering (optimize k using elbow)

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

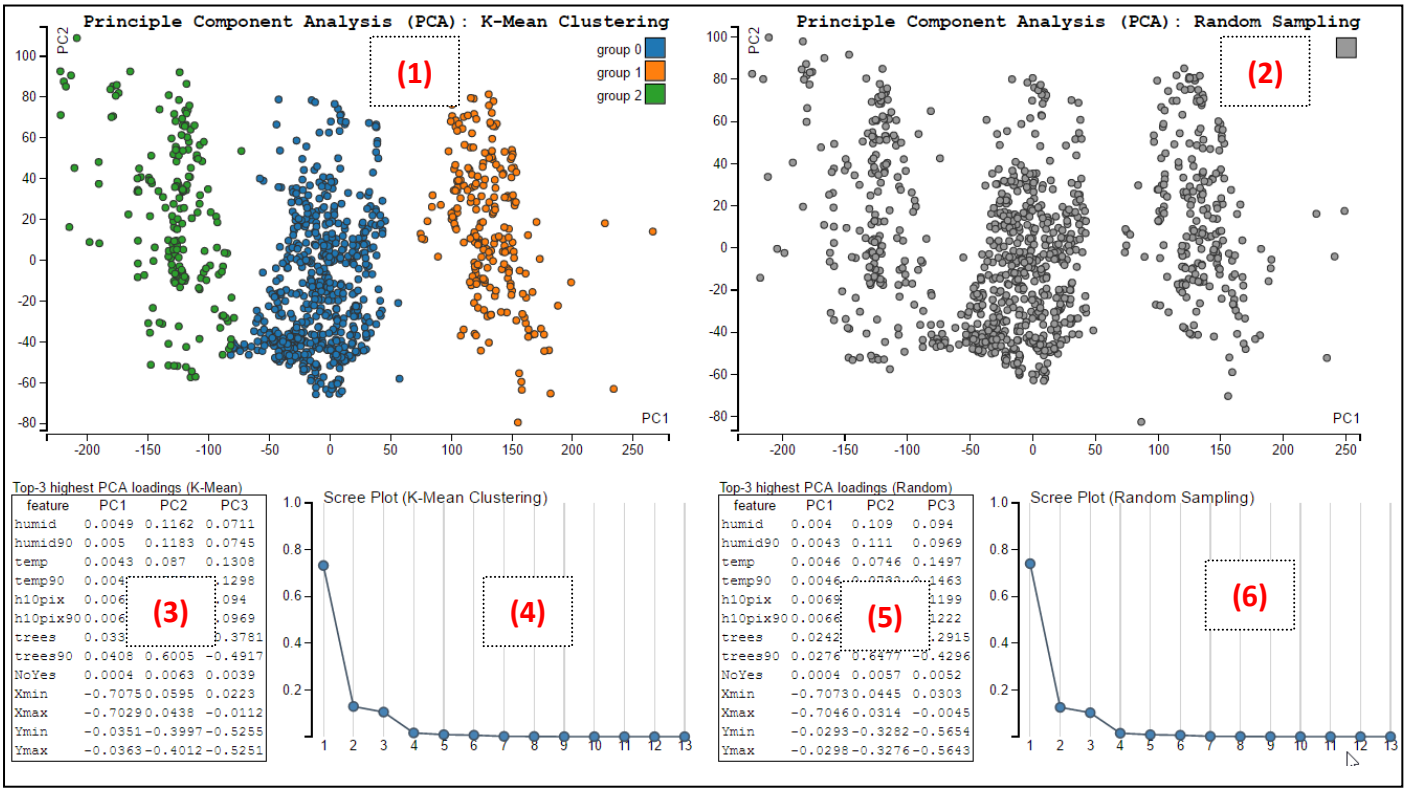
- find the intrinsic dimensionality of the data using PCA
- produce scree plot visualization and mark the intrinsic dimensionality
- obtain the three attributes with highest PCA loadings

**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
- visualize scatterplot matrix of the three highest PCA loaded attributes

3. Layout

   A. Principle Component Analysis (PCA):



Components:

(1) Scatter plot of data projected into the top two PCA vectors (Adaptive sampling + K-Mean clustering)

(2) Scatter plot of data projected into the top two PCA vectors (Random sampling)

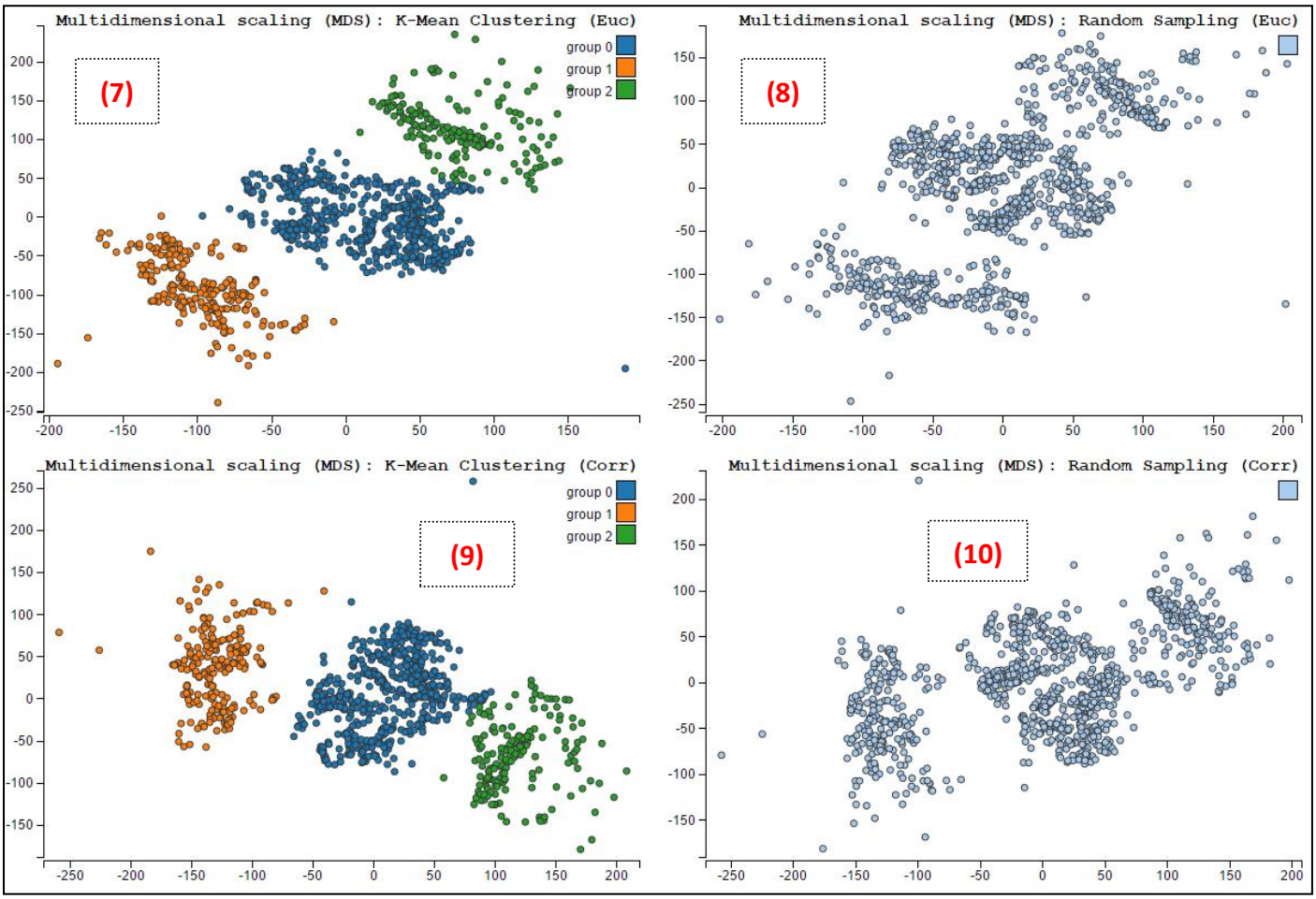
(3) Matrix of the three highest PCA loaded attributes (Adaptive sampling + K-Mean clustering)

(4) Scree plot to show all eigenvalues in PCA (Adaptive sampling + K-Mean clustering)

(5) Matrix of the three highest PCA loaded attributes (Random sampling)

(6) Scree plot to show all eigenvalues in PCA (Random sampling)

B. Multi-Dimensional Scaling (MDS):




Components:

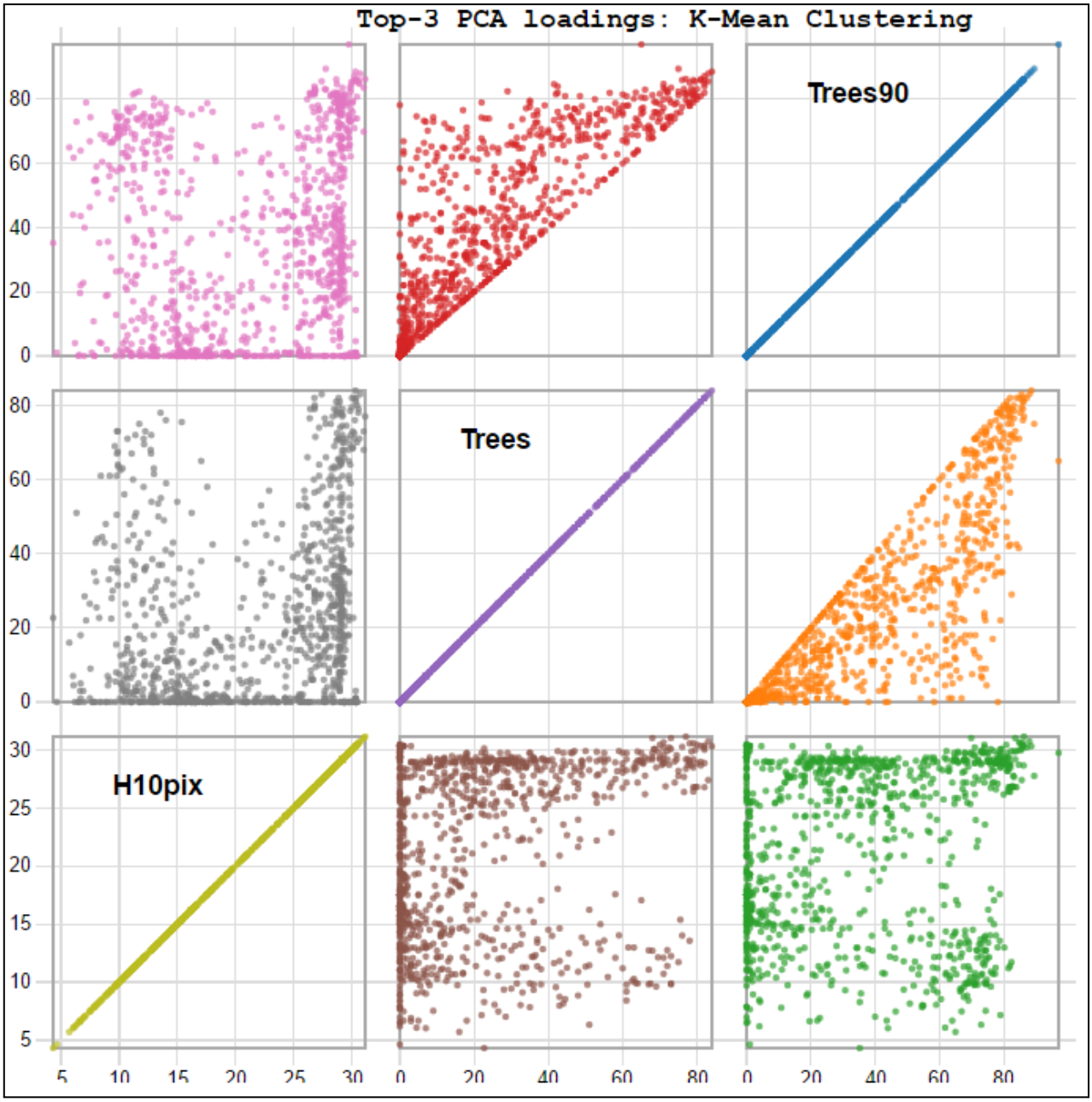
(7) Scatter plot of data after MDS with Euclidian distance (Adaptive sampling + K-Mean clustering)

(8) Scatter plot of data after MDS with Euclidian distance (Random sampling)

(9) Scatter plot of data after MDS with Correlation distance (Adaptive sampling + K-Mean clustering)

(10) Scatter plot of data after MDS with Correlation distance (Random sampling)

C. Scatter plot matrix of top-3 PCA loadings:




Components:

(11) Scatter plot matrix of Top-3 PCA loadings (K-Mean clustering)

4. Code Implementation in Python

A. Data Source: https://vincentarelbundock.github.io/Rdatasets/doc/DAAG/dengue.html -> Dengue fever

B. Pre-process data

i. Import csv file

```
# import csv file
with open('dengue.csv', 'r') as f:
    reader = csv.reader(f)
    inFile = list(reader)
    rowName = inFile[0][:]
    inFile = inFile[1:][:]
```

ii. Remove invalid values in the original data

```
# All input data
# data pre-proccessing: remove "NA" items by replacing average values
for i in range(len(inFile[0])):
    avg = 0.0
    cnt = 0.0
    for j in range(len(inFile)):
        if inFile[j][i] != 'NA':
            cnt += 1
            avg += float(inFile[j][i])
    avg = avg / cnt
    for j in range(1, len(inFile)):
        if inFile[j][i] == 'NA':
            inFile[j][i] = avg
inputData = [inFile[i][1:] for i in range(1, len(inFile))]
```

C. Data reduction

i. Random sampling and Adaptive sampling with K-Mean clustering

ii. Write the output data after sampling into a csv file

```
# Task (1a) Random-sampling: 2000 -> 1000
randInputData = [inputData[i] for i in random.sample(range(len(inputData)), sample_size)]

# Task (1b) K-means clustering + Adeptive Sampling
kmLabels = KMeans(n_clusters = nGroup).fit(inputData).labels_

# Seperate input data into n groups
kmInputData = np.append(np.asarray(inputData), np.asarray(kmLabels).reshape((len(kmLabels), 1)), 1)
kmInputData = kmInputData.tolist()

# seperate input data into n groups
inputDataGroup = [[] for i in range(nGroup)]
for i in range(len(kmInputData)):
    inputDataGroup[kmLabels[i]].append(kmInputData[i])

# Adeptive Sampling: 2000 -> 1000
kmInputData = []
for i in range(nGroup):
    kmInputData.extend([inputDataGroup[i][j] for j in random.sample(range(len(inputDataGroup[i])),
        int(len(inputDataGroup[i]) * sample_size / len(inputData)))])
```

D. Dimension reduction - PCA

i. Perform PCA

ii. Obtain the three attributes with highest PCA loadings (for Top-3 PCA loadings)

iii. Obtain all eigenvalues (for scree plot)

iv. Obtain the values projected into the top two PCA vectors (for scatter plot)

v. Write the output data of PCA into a csv file

```
randpca = PCA().fit(randInputData)
pcaRandComponents = randpca.components_[0:3]
pcaRandEigenvalue = randpca.explained_variance_ratio_

with open('Random_PCA_dengue_intrinsic_dimensionality.csv', 'w', newline = '') as f:
    writer = csv.writer(f)
    writer.writerows([['feature','PC1','PC2','PC3', 'ev']])
    pcaRandComponents = np.append(rowName, pcaRandComponents, 0)
    pcaRandComponents = np.append(pcaRandComponents, pcaRandEigenvalue.reshape(1,len(pcaRandEigen
    writer.writerows(np.transpose(pcaRandComponents))

pcaRandOutputData = PCA(n_components=2).fit_transform(randInputData)
pcaRandOutputData = np.append(pcaRandOutputData, randInputData, 1)
with open('Random_PCA_dengue.csv', 'w', newline = '') as f:
    writer = csv.writer(f)
    writer.writerows(np.append([['x','y']], rowName, 1))
    writer.writerows(pcaRandOutputData)
```

E. Dimension reduction - MDS

i. Perform MDS with Euclidean distance

```
## Task 3(a) euclidean_distances

randInputData = np.asarray(randInputData)
mdsRandEucOutputData = MDS(n_components=2).fit(randInputData).embedding_
mdsRandEucOutputData = np.append(mdsRandEucOutputData, randInputData, 1)

with open('Random_MDS_Euc_dengue.csv', 'w', newline = '') as f:
    writer = csv.writer(f)
    writer.writerows(np.append([['x','y']], rowName, 1))
    writer.writerows(mdsRandEucOutputData)
```

ii. Perform MDS with correlation distance

```
## Task 3(b) correlation_distances

similarities = euclidean_distances(randInputData)
mdsRandCorrOutputData = MDS(n_components=2, dissimilarity="precomputed").fit(similarities).embedding_
mdsRandCorrOutputData = np.append(mdsRandCorrOutputData, randInputData, 1)

with open('Random_MDS_Corr_dengue.csv', 'w', newline = '') as f:
    writer = csv.writer(f)
    writer.writerows(np.append([['x','y']], rowName, 1))
    writer.writerows(mdsRandCorrOutputData)
```

5. Code Implementation in D3

A. Create all SVG elements

```
var svgKMeanPCA = d3.select("body")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

var svgRandPCA = d3.select("body")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

B. Load csv output data

```
// load data KMean output
d3.csv("KMean_PCA_dengue.csv", function(error, data) {

    // change string into number format
    data.forEach(function(d) {
        d.x = +d.x;
        d.y = +d.y;
        d.group = +d.group;
    });
```

C. Scale the range of output data

```
    // Scale the range of input data
    xScale.domain([d3.min(data, xValue) * 1.05, d3.max(data, xValue)* 1.05]);
    yScale.domain([d3.min(data, yValue) * 1.05, d3.max(data, yValue)* 1.05]);
```

D. Draw scatter plots and add d3 mouse events: MouseOver, MouseOut

```
218     // draw points
219     svgKMeanPCA.selectAll(".point")
220         .data(data)
221         .enter()
222         .append("circle")
223         .attr("class", "point")
224         .attr("r", 3)
225         .attr("cx", function(d) { return xScale(xValue(d)); })
226         .attr("cy", function(d) { return yScale(yValue(d)); })
227         .style("stroke", "#333")
228         .style("fill", function(d) { return color(cValue(d));})
229         .on("mouseover", function(d) {
230             svgKMeanTooltip.transition().duration(200).style("opacity", .9);
231             svgKMeanTooltip.html("(" + (Math.floor(xValue(d) * 10000) / 10000) + "," +
232                 (Math.floor(yValue(d) * 10000) / 10000) +
233                 ") <br/> humid: " + (Math.floor(d.humid * 100) / 100) +
234                 "<br/> temperature: " + (Math.floor(d.temp * 100) / 100) + "&#8451" +
235                 "<br/> Tree coverage: " + (Math.floor(d.trees * 100) / 100) + "%" +
236                 "<br/> longitude: " + (Math.floor(d.Xmin * 10000) / 10000) + " ~ " +
237                 (Math.floor(d.Xmax * 10000) / 10000) + "<br/> longitude: " +
238                 (Math.floor(d.Ymin * 10000) / 10000) + " ~ " + (Math.floor(d.Ymax * 10000)
239                 .style("left", (d3.event.pageX + 5) + "px")
240                 .style("top", (d3.event.pageY - 30) + "px");
241 
242         })
243         .on("mouseout", function(d) {
244             svgKMeanTooltip.transition().duration(500).style("opacity", 0);
245         });
```

MouseOver event:

Show detail information of the selected point on the graph

MouseOut event:

Remove the information

E. Load csv output data or scree plots

```
369 d3.csv("KMean_PCA_dengue_intrinsic_dimensionality.csv", function(error, data) {
370 
371     var EigenValue = [];
372     for (var i = 0; i < data.length; i++) {
373         EigenValue.push(+data[i].ev);
374     }
375 
376     data.forEach(function(d) {
377         d.PC1 = Math.round(+d.PC1 * 10000) / 10000;
378         d.PC2 = Math.round(+d.PC2 * 10000) / 10000;
379         d.PC3 = Math.round(+d.PC3 * 10000) / 10000;
380     });
381 
382     screeX.domain([1, data.length]);
383     screeY.domain([1e-6, 1]);
```

F. Draw points and lines on the scree plot

```
405     svgKMeanScreePlot.selectAll(".point")
406         .data(EigenValue)
407         .enter()
408         .append("circle")
409         .attr("class", "point")
410         .attr("r", 4)
411         .attr("cx", function(d,i) { return screeX(i+1); })
412         .attr("cy", function(d) { return screeY(d); })
413         .style("stroke", "#456")
```

```
    svgKMeanScreePlot.append("path")
        .attr("d", line(EigenValue))
        .style("stroke", "#456")
        .style("stroke-width", 1.5)
        .style("fill", "none");
```

G. Build a table to show the top-3 PCA loadings

```
// The table generation function
var tabulate = function(data, columns, table) {
    table.append("thead").append("tr")
        .selectAll("th")
        .data(columns)
        .enter()
        .append("th")
        .text(function(column) { return column; });

    var rows = table.append("tbody").selectAll("tr").data(data).enter().append("tr");

    var cells = rows.selectAll("td")
        .data(function(row) {
            return columns.map(function(column) {
                return {column: column, value: row[column]};
            });
        })
        .enter()
        .append("td")
        .attr("style", "font-family: Courier")
        .html(function(d) { return d.value; })
        .on("mouseover", function(d) {
            d3.select(this).attr("style", "outline: thin solid red; font: bold 12px C
        })
        .on("mouseout", function(d) {
            d3.select(this).attr("style", "color: black;")
                .attr("style", "font-family: Courier")
        });
}
```

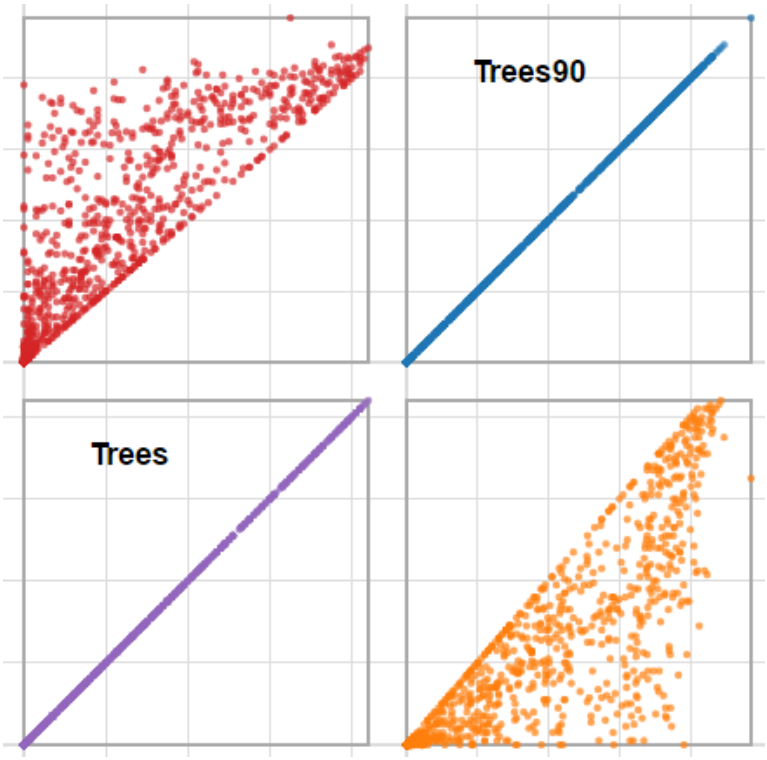
H. Scatter plots for all the results in MDS

```
d3.csv("Random_MDS_Euc_dengue.csv", function(error, data) {

    data.forEach(function(d) {
        d.x = +d.x;
        d.y = +d.y;
    });

    xScale.domain([d3.min(data, xValue) * 1.05, d3.max(data, xValue)* 1.05]);
    yScale.domain([d3.min(data, yValue) * 1.05, d3.max(data, yValue)* 1.05]);

    svgRandMDSEuc.append("g")
        .attr("class", "axis")
        .attr("transform", "translate(0," + (height * 1.01) + ")")
        .call(xAxis);

    svgRandMDSEuc.append("g")
        .attr("class", "axis")
        .call(yAxis);
```

6. How to setup Python server:

Command Line: " python –m http.server 8000 "
Open your browser with http://your_ip_address:8000/

7. Discussion:
   A. The grouping results via K-Means clustering are good. From the scatter plots of PCA, we can found points of different color are almost separated into three groups. (K = 3)
   B. In the scatter plot matrix, the attributes top-3 PCA loadings are "Trees", "Trees90", "H10pix". It is obvious that the value in "Trees" and "Trees90" are correlated, so the figure shows a triangle instead of random plot on the figure.




   C. MDS takes much longer processing time than PCA. In one round, PCA takes about 3-5 seconds while MDS take about 1 minute.