LAB 2

DATASET

Chosen Dataset: **Benefits, Unemployment of Blue Collar Workers**

Link to Dataset: https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/benefits.csv

Dataset contains 4877 rows & 18 dimensions. Some of them are numerical. I have changed the remaining categorical attributes into numerical attributes as well.

4	stateur	statemb	state	age	tenure	joblost	nwhite	school12	sex	bluecol	smsa	married	dkids	dykids	yrdispl	rr	head	ui
1	4.5	167	42	49	21	other	no	no	male	yes	yes	no	no	no	7	0.290631	yes	yes
2	10.5	251	55	26	2	slack_work	no	no	male	yes	yes	no	yes	yes	10	0.520202	yes	no
3	7.2	260	21	40	19	other	no	yes	female	yes	yes	yes	no	no	10	0.4324895	yes	yes
4	5.8	245	56	51	17	slack_work	yes	no	female	yes	yes	yes	no	no	10	0.5	no	yes
5	6.5	125	58	33	1	slack_work	no	yes	male	yes	yes	yes	yes	yes	4	0.390625	yes	no
6	7.5	188	11	51	3	other	no	no	male	yes	yes	yes	no	no	10	0.4822007	yes	yes
7	5.8	166	93	30	5	position_abc	no	no	male	yes	yes	yes	yes	yes	6	0.3340425	yes	yes
8	5.8	214	84	26	3	slack_work	no	no	male	yes	yes	yes	yes	yes	8	0.510989	yes	yes
9	7.7	213	84	54	20	other	no	no	male	yes	yes	yes	no	no	6	0.3556338	yes	no
10	6	187	33	31	1	slack_work	no	no	male	yes	yes	yes	no	no	8	0.3732719	yes	yes
11	7.1	206	33	56	20	slack_work	no	no	male	yes	no	yes	no	no	10	0.4044665	yes	yes
12	10.1	132	12	54	26	other	no	no	male	yes	no	yes	yes	no	2	0.4132948	yes	no
13	5	279	22	21	1	slack_work	no	no	male	yes	yes	no	no	no	9	0.5984849	no	no
14	6.6	291	22	35	1	slack_work	yes	no	male	yes	yes	no	yes	yes	10	0.5991561	no	no
15	7.2	166	93	23	1	slack_work	no	yes	male	yes	yes	no	no	no	4	0.39375	no	yes
16	7.2	260	21	41	3	other	no	no	male	yes	yes	yes	yes	no	10	0.5023041	no	yes
17	6.4	213	84	49	12	other	no	no	male	yes	yes	yes	no	no	7	0.4263736	no	yes
18	8.9	166	56	35	1	other	yes	no	male	yes	no	yes	no	no	2	0.473545	yes	no
19	5.8	245	56	40	12	other	no	no	female	yes	yes	yes	yes	no	10	0.4981949	no	yes
20	6.1	138	54	60	30	slack_work	no	no	male	yes	yes	yes	no	no	2	0.5185185	yes	no
21	7.2	260	21	42	4	slack_work	yes	no	female	yes	yes	yes	no	no	10	0.5	no	yes
22	18	211	55	48	3	other	no	no	male	yes	no	yes	no	no	2	0.2558922	yes	yes
23	4.6	168	16	41	3	other	no	no	male	yes	yes	yes	no	no	3	0.3359073	yes	yes
24	9	192	83	29	5	position_abc	no	no	male	yes	no	yes	yes	yes	5	0.2622036	yes	no
25	5.9	185	44	59	5	slack_work	no	no	male	yes	no	yes	no	no	4	0.5	yes	yes
26	7.4	197	73	32	3	slack_work	no	no	male	yes	no	yes	yes	yes	6	0.4389671	yes	yes
27	11.4	168	33	22	3	slack_work	no	no	male	yes	yes	no	yes	no	2	0.504065	no	no
28	5.5	150	43	25	3	slack_work	yes	no	male	yes	yes	yes	yes	yes	8	0.5752212	no	no
29	4.8	179	44	36	3	other	no	yes	male	yes	no	yes	yes	yes	7	0.4010989	yes	yes
30	6.3	150	59	41	20	other	no	no	male	yes	yes	no	yes	no	3	0.3994845	no	yes
31	5	138	54	26	2	slack_work	no	no	male	yes	yes	no	no	no	3	0.5204461	no	yes
32	11.5	110	62	25	3	other	no	no	female	yes	yes	no	yes	yes	2	0.4175258	yes	no
33	4.9	194	88	23	1	slack_work	no	no	male	yes	yes	no	yes	yes	9	0.5194175	no	yes
34	7.7	147	74	24	1	slack_work	no	no	male	yes	no	yes	no	no	1	0.4593838	no	no
35	8.8	197	34	22	3	slack_work	no	no	male	yes	yes	yes	yes	yes	5	0.4872611	yes	yes
36	7.9	151	61	38	3	other	no	no	female	yes	no	no	no	no	7	0.6147541	yes	yes
37	3.9	176	54	37	10	other	yes	no	female	yes	yes	yes	yes	yes	8	0.5032895	no	yes
38	9.2	276	34	39	3	other	no	no	male	yes	yes	yes	yes	no	10	0.5253165	yes	no

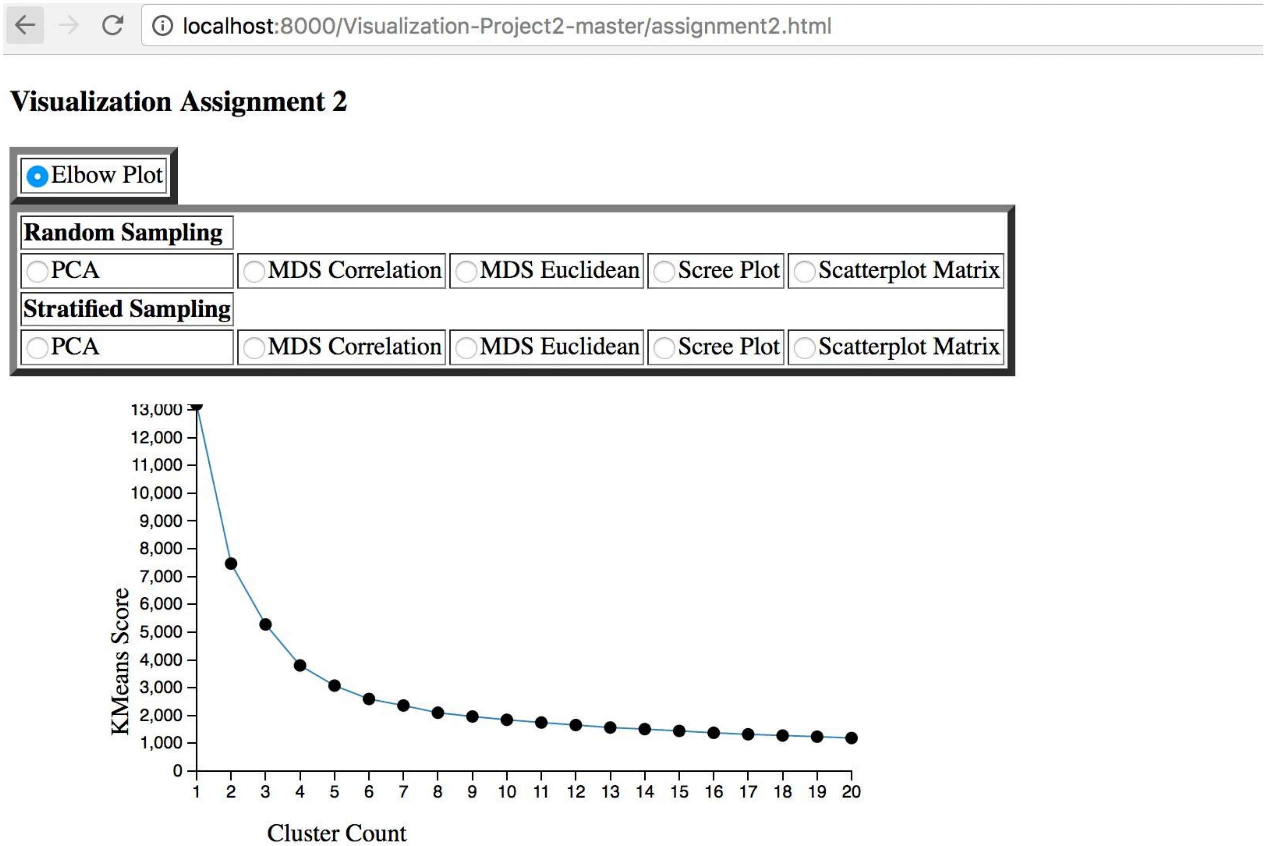
I have used client server system: python for processing (server), D3 (v3) for VIS (client).

TASKS AND IMPLEMENTATION

Task1: data clustering and decimation (30 points)

I used Python's numpy' random choice function to randomly select 20% 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 20% from each of the four clusters.

K means Elbow Plot



Code Snippets

```python
def adaptive_sampling(data_frame, cluster_count,fraction):
    k_means = Kcluster.KMeans(n_clusters=cluster_count)
    print ('k_means before',k_means)
    k_means.fit(data_frame)
    print ('k_means after',k_means)
    data_frame['label'] = k_means.labels_
    print ('label',k_means.labels_)
    adaptiveSampleRows = []
    for i in range(cluster_count):
        adaptiveSampleRows.append(data_frame.ix[random.sample(data_frame[data_frame['label'] == i].index, (int)
            (len(data_frame[data_frame['label'] == i])*fraction))])

    adaptiveSample = pd.concat(adaptiveSampleRows)
    del adaptiveSample['label']

return adaptiveSample


def random_sampling(data_frame, fraction):
    rows = random.sample(data_frame.index, (int)(len(data_frame)*fraction))
    return data_frame.ix[rows]
```

```python
def find_Elbow_Point(data_frame):
    sse = []
    maxK = 18
    minInertia = 10000000000000
    minK = 0
    kArr = list(xrange(21))
    kArr.pop(0)
    inertiaArr= []
    print kArr

    for k in range(1,21):
        # sse[k] = 0
        k_means = Kcluster.KMeans(n_clusters=k)
        k_means.fit(data_frame)
        clusters = k_means.labels_
        # print ('clusters', clusters)
        inertias = k_means.inertia_
        inertiaArr.append(inertias)
        if(minInertia>inertias):
            minInertia = inertias
            minK = k

        print ('inertia',inertias)
        # print('inertia',inertias/k)
    print ("minK ",minInertia,minK)

    # plt.plot( kArr, inertiaArr,c='blue',label="test1")
    # plt.xticks(np.arange(1, 21, 1.0))
    # plt.show()
    elbowDF =  pd.DataFrame({'KMeans_Score' : inertiaArr})
    elbowDF['Cluster_Count'] = kArr
    elbowDF.to_csv('elbow.csv', sep=',')#adaptivescreeplt
```

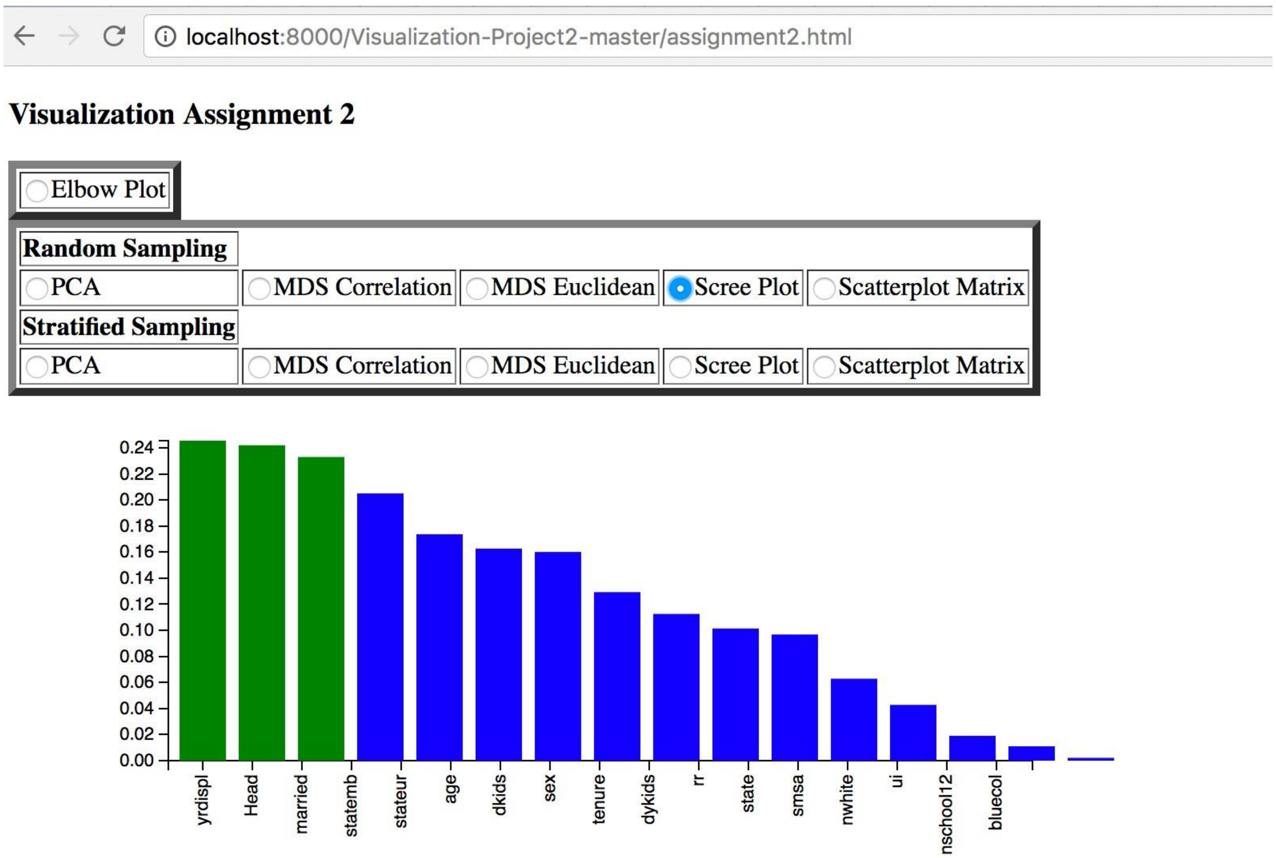
Task 2: dimension reduction (use decimated data) (30 points)

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.

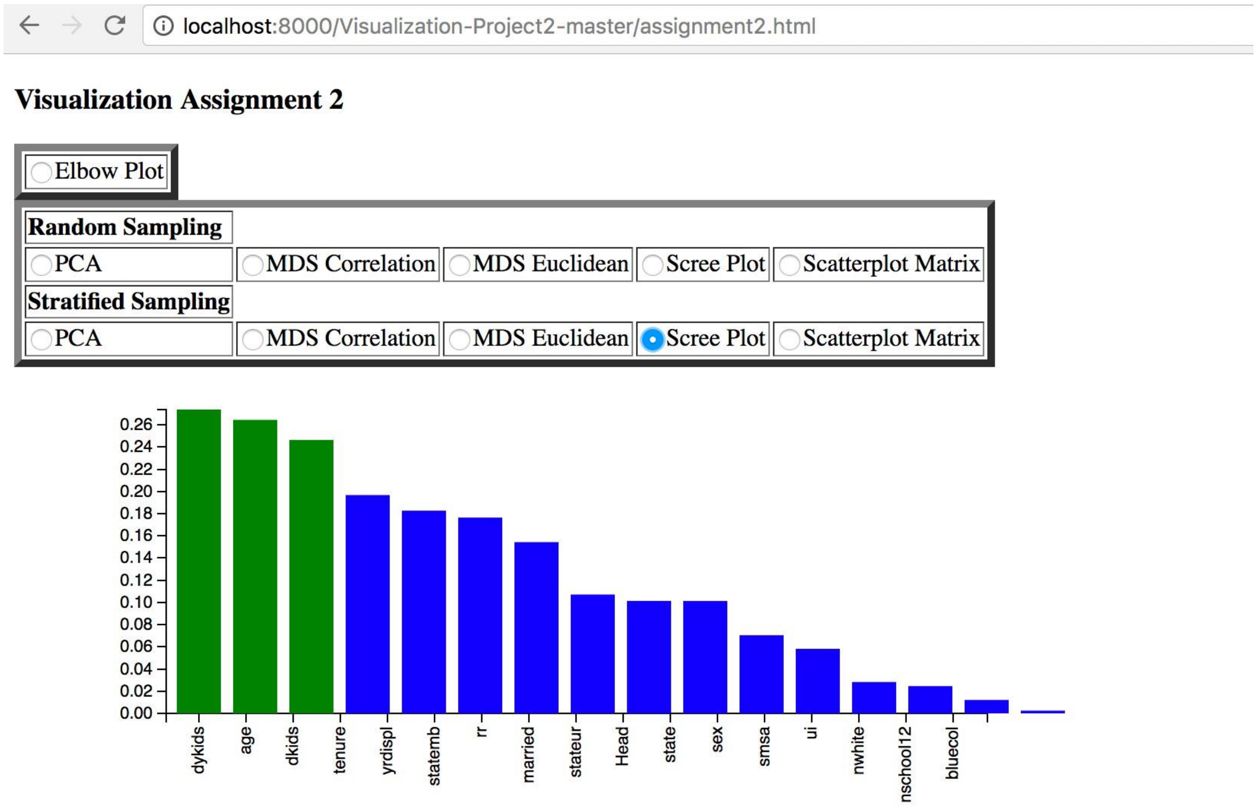
Scree Plots for Random and Stratified Samples

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

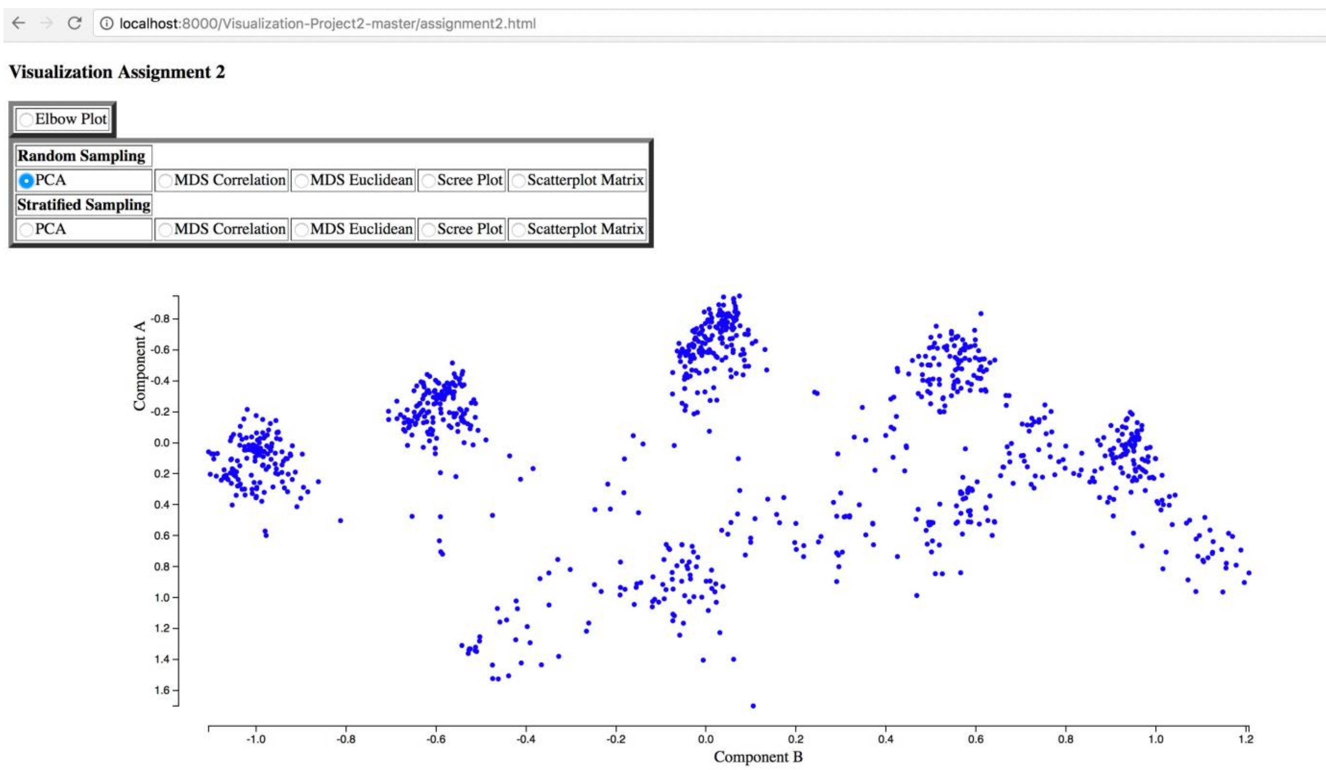


To obtain the three highest attributes with maximum PCA loading, data is first reduced to only the intrinsic dimensions and then I found the sum of squared loading value for all the attributes. Attributes with the highest sum of this squared value were considered as the three attributes with maximum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

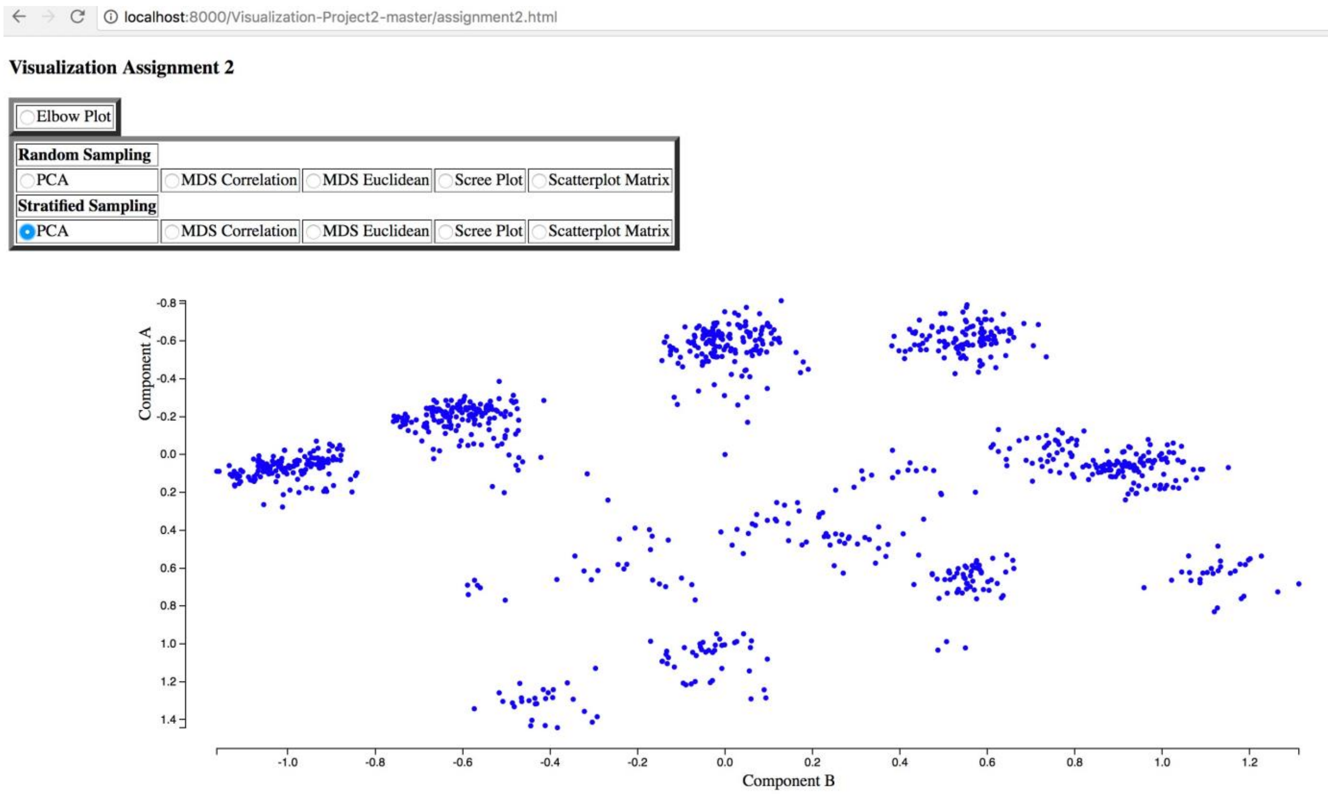
Task 3: visualization (use dimension reduced data) (40 points)

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

PCA (Random Sampling)

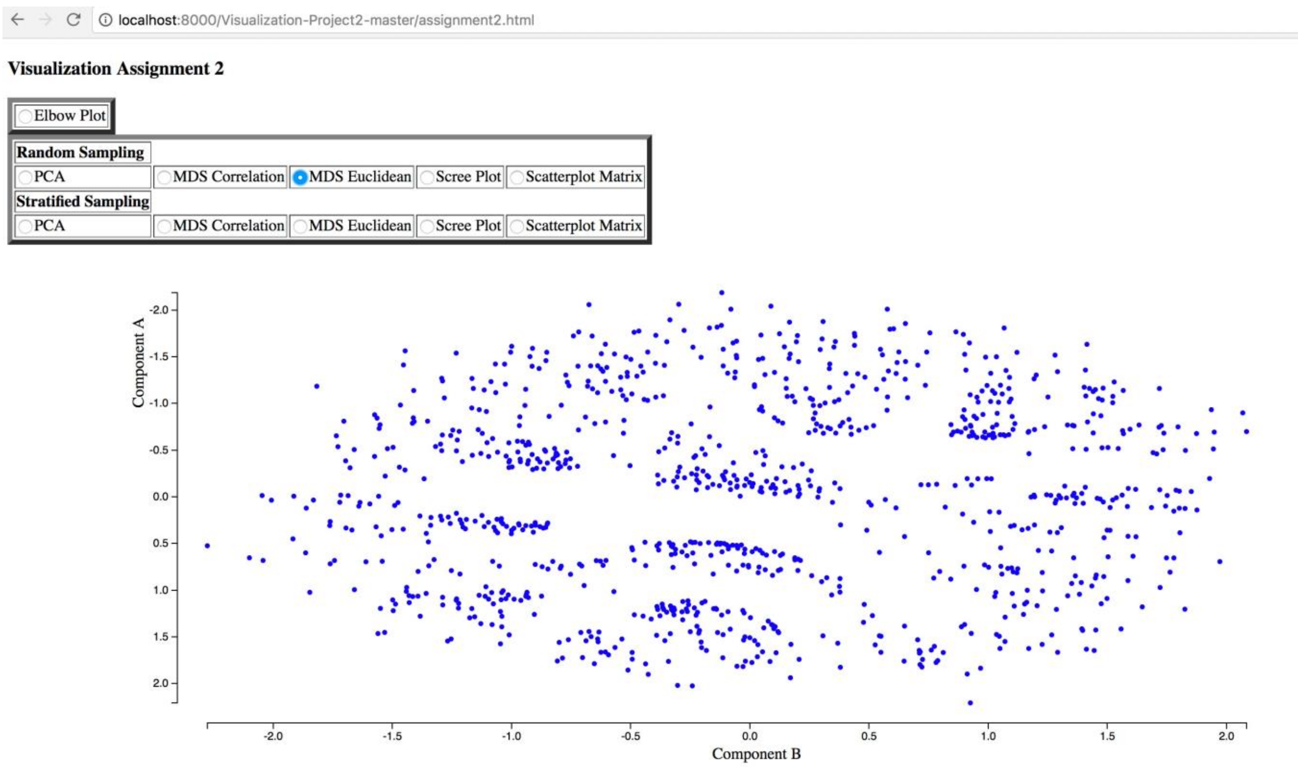


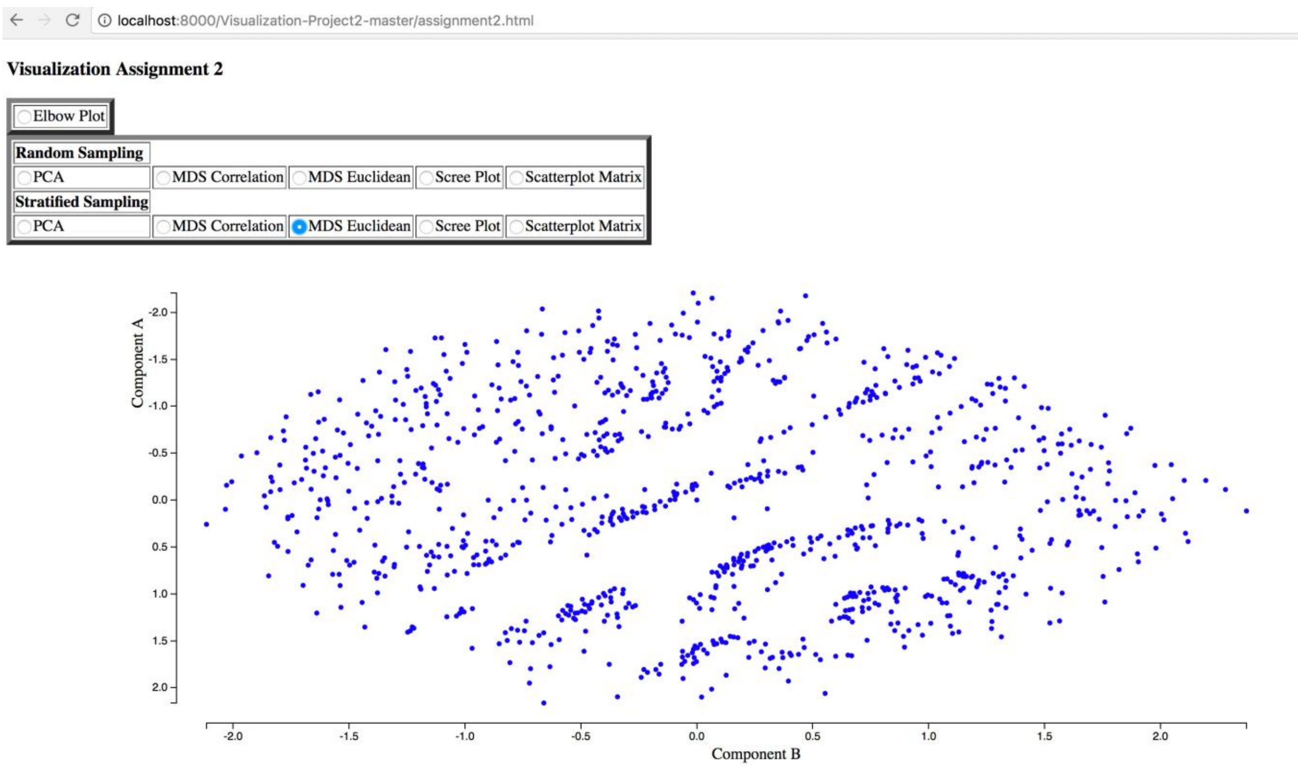
PCA (Stratified Sampling)



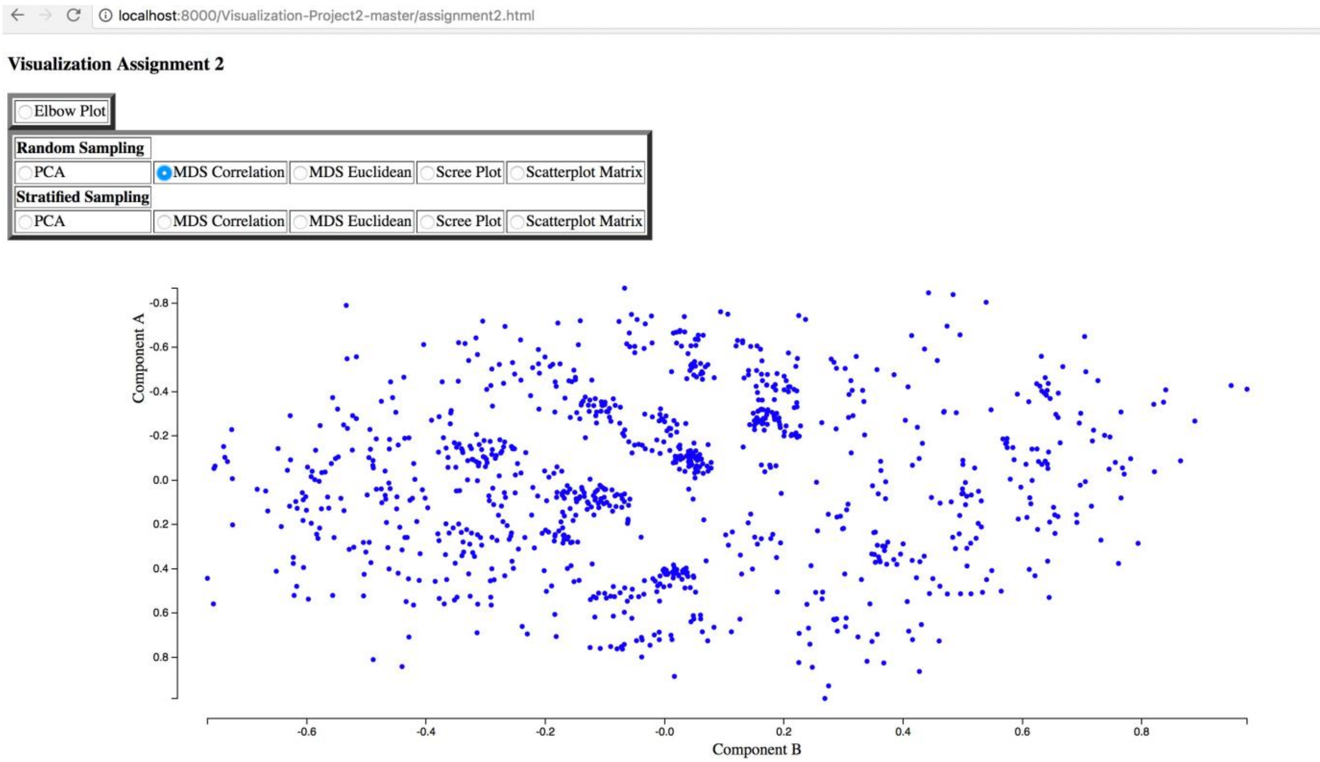
Visualize data via MDS (Euclidian & correlation distance) in 2D scatterplots

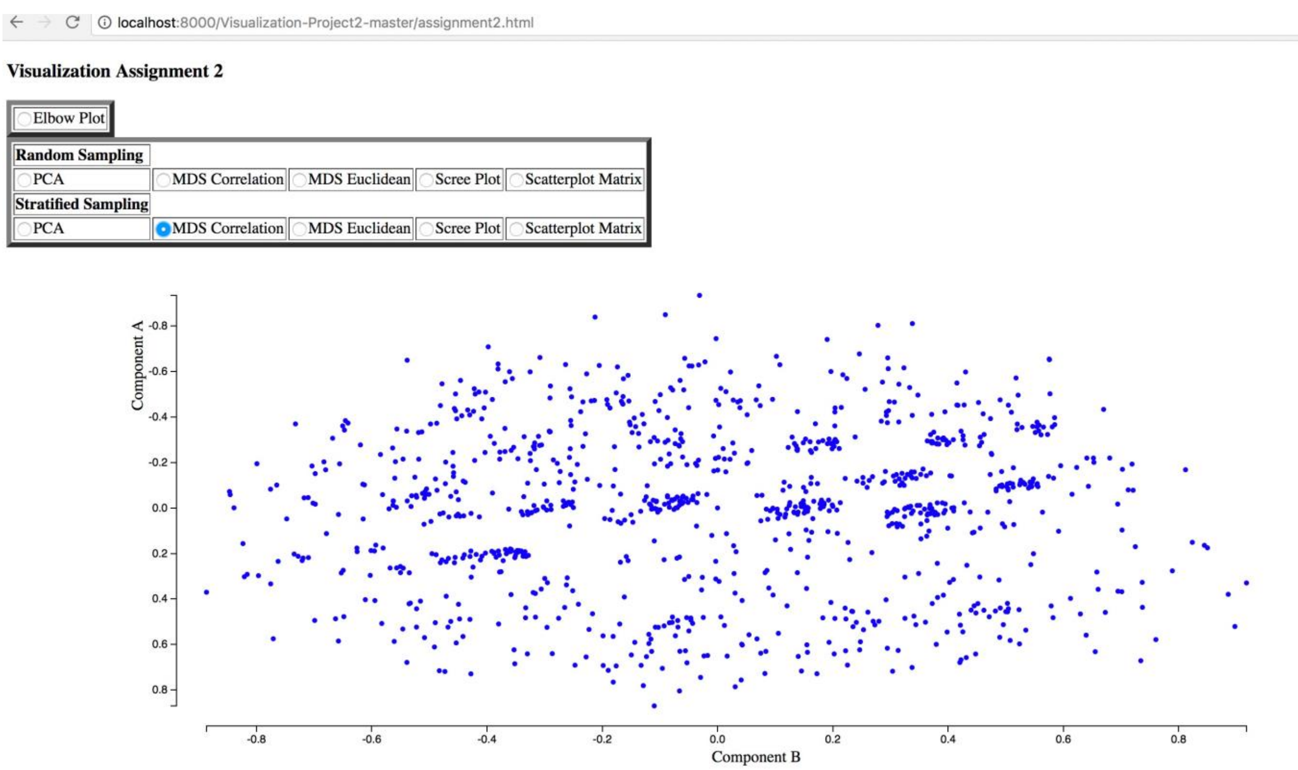
Euclidean




localhost:8000/Visualization-Project2-master/assignment2.html
Visualization Assignment 2
Elbow Plot
Random Sampling
PCA
MDS Correlation
MDS Euclidean
Scree Plot
Scatterplot Matrix
Stratified Sampling
PCA
MDS Correlation
MDS Euclidean
Scree Plot
Scatterplot Matrix
Component A
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
Component B
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0

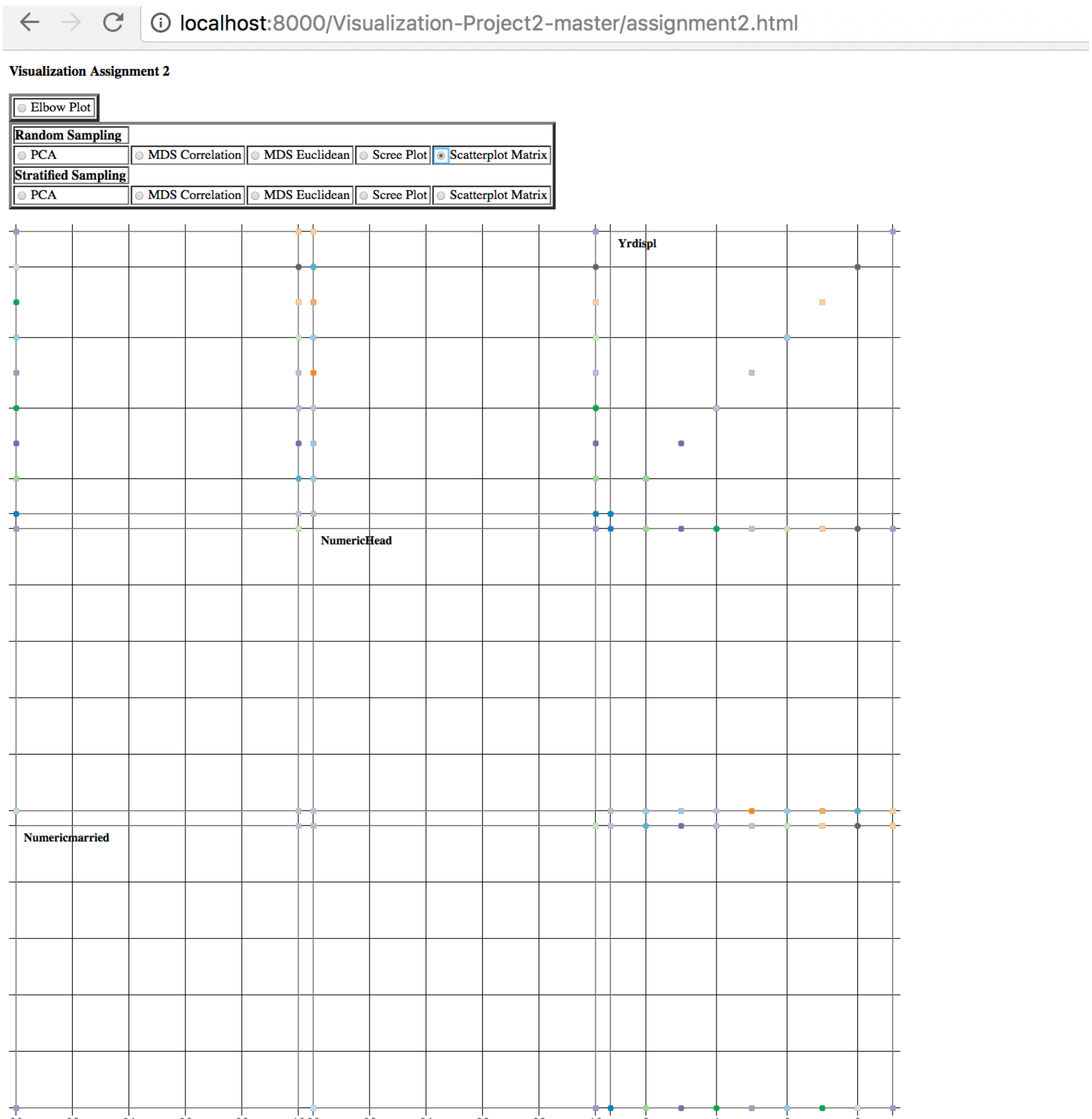
Correlation



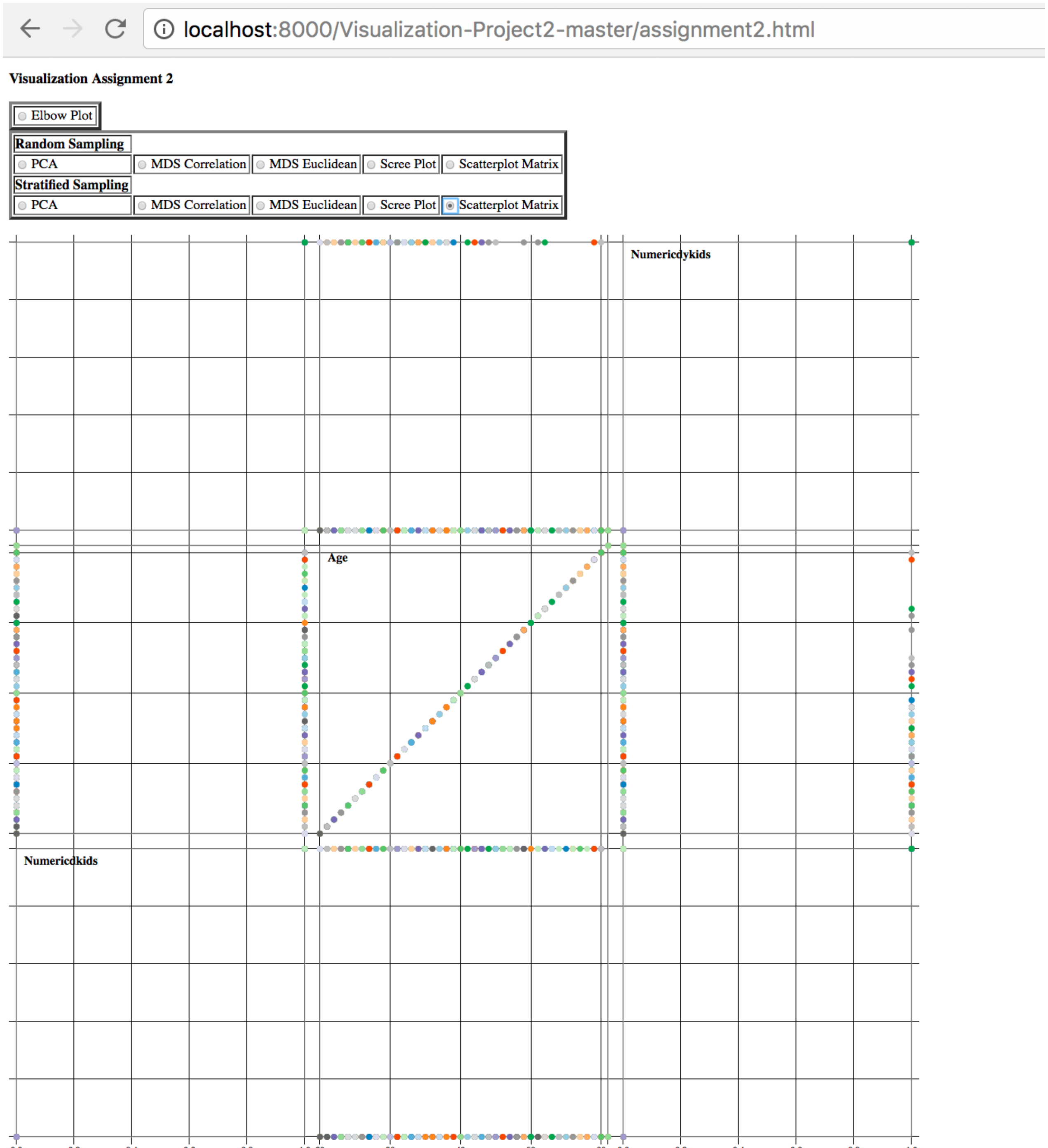


Visualize scatterplot matrix of the three highest PCA loaded attributes

Random Sampling



Startified Sampling



Code Snippets

elbowplot.js

```
// Define the line
var valueline = d3.svg.line()
                .y(function(d) { return y(d.KMeans_Score/1000); })
                .x(function(d) { return x(d.Cluster_Count); });
// Get the data
d3.csv("./data/elbow.csv", function(error, data) {
    data.forEach(function(d) {
        d.KMeans_Score = +d.KMeans_Score;
        d.Cluster_Count = +d.Cluster_Count;
    });

    // Scale the range of the data
    y.domain([0,d3.max(data, function(d) { return d.KMeans_Score/1000; }) ]);
    x.domain(d3.extent(data, function(d) { return d.Cluster_Count; }));

    // Add the valueline path.
    svg.append("path")
        .attr("class", "line")
        .attr("d", valueline(data))
        .attr("transform", "translate(120,0)");

    svg.selectAll("dot")
        .data(data)
        .enter().append("circle")
        .attr("transform", "translate(120,0)")
        .attr("r", 4)
        .attr("cx", function(d) { return x(d.Cluster_Count); })
        .attr("cy", function(d) { return y(d.KMeans_Score/1000); });

    // Add the X Axis
    svg.append("g")
        .attr("class", "x axis")
        .attr("transform", "translate(120," + height+ ")")
        .call(xAxis);

    // Add the Y Axis
    svg.append("g")
        .attr("class", "y axis")
        .attr("transform", "translate(120,0)")
        .call(yAxis);

    // Add the text label for the Y axis
    svg.append("text")
        .attr("transform", "rotate(-90)")
        .attr("y", left_pad-140)
        .attr("x",height-400)
        .attr("dy", "1em")
        .style("text-anchor", "middle")
        .text("KMeans Score");

    svg.append("text")
        .attr("transform", "translate(" + (width / 2) + " ," + (height + margin.bottom-5) + ")")
        .style("text-anchor", "middle")
        .text("Cluster Count");

    svg.append("circle")
        .data(data)
        .enter()
        .append("circle")
        .attr("r", 3)
        .attr("cx", function(d){
            return x(d.Cluster_Count);
        })
        .attr("cy", function(d){
            return y(d.KMeans_Score/1000);
        })
```

Scatterplot.js (Random sampling)

elbowplot.js | loadings1.js | scatter_plot.js | scatterplot_matrix.js | screeplot.js

```
}

function plot_values(filename) {

    filename = "./data/" + filename;
    svg.selectAll("*").remove();

    // Load data
    d3.csv(filename, function(error, data) {
        data.forEach(function(d) {
            d.r1 = +d.r1;
            d.r2 = +d.r2;

        });

        var xValueR = function(d) { return d.r1;};
        var yValueR = function(d) { return d.r2;};

        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-10)+")")
            .call(xAxis);

        svg.append("g")
            .attr("class", "axis")
            .attr("transform", "translate("+(left_pad-pad)+", 0)")
            .call(yAxis);

        svg.append("text")
            .attr("transform", "rotate(-90)")
            .attr("y", left_pad-80)
            .attr("x",h-600)
            .attr("dy", "1em")
            .style("text-anchor", "middle")
            .text("Component A");

        svg.append("text")
            .attr("y", h-20)
            .attr("x", h+250)
            .attr("dy", "1em")
            .style("text-anchor", "middle")
            .text("Component B");

        svg.selectAll("circle")
            .data(data)
            .enter()
            .append("circle")
            .attr("r", 2.5)
            .attr("cx", function(d){
                return xScale(d.r1);
            })
            .attr("cy", function(d){
                return yScale(d.r2);
            })
            .style("fill", "blue");
    });

}
```

Scatter_plot.js (Stratified Sampling)

elbowplot.js loadings1.js scatter_plot.js scatterplot_matrix.js screeplot.js

```
}

function plot_values(filename) {

    filename = "./data/" + filename;
    svg.selectAll("*").remove();

    // Load data
    d3.csv(filename, function(error, data) {
        data.forEach(function(d) {
            d.r1 = +d.r1;
            d.r2 = +d.r2;

        });

        var xValueR = function(d) { return d.r1;};
        var yValueR = function(d) { return d.r2;};

        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-10)+")")
            .call(xAxis);

        svg.append("g")
            .attr("class", "axis")
            .attr("transform", "translate("+(left_pad-pad)+", 0)")
            .call(yAxis);

        svg.append("text")
            .attr("transform", "rotate(-90)")
            .attr("y", left_pad-80)
            .attr("x",h-600)
            .attr("dy", "1em")
            .style("text-anchor", "middle")
            .text("Component A");

        svg.append("text")
            .attr("y", h-20)
            .attr("x", h+250)
            .attr("dy", "1em")
            .style("text-anchor", "middle")
            .text("Component B");

        svg.selectAll("circle")
            .data(data)
            .enter()
            .append("circle")
            .attr("r", 2.5)
            .attr("cx", function(d){
                return xScale(d.r1);
            })
            .attr("cy", function(d){
                return yScale(d.r2);
            })
            .style("fill", "blue");
    });

}
```

screeplot.js

elbowplot.js   loadings1.js   scatter_plot.js   scatterplot_matrix.js   screeplot.js

```
function scree_plot(filename) {

    filename = "./data/" + filename;
    svg.selectAll("*").remove();

    // Set the dimensions of the canvas / graph
    var margin = {top: 30, right: 20, bottom: 50, left: 50},
        width = 800 - margin.left - margin.right,
        height = 400 - margin.top - margin.bottom;

    // Set the ranges
    var x = d3.scale.linear().range([0, width]);
    var y = d3.scale.linear().range([height, 0]);

    // Define the axes
    var xAxis = d3.svg.axis().scale(x)
        .orient("bottom").ticks(18);

    var yAxis = d3.svg.axis().scale(y)
        .orient("left").ticks(7);

    // Define the line
    var valueline = d3.svg.line()
        .y(function(d) { return y(d.eigen_value); })
        .x(function(d) { return x(d.PCA_components); });

    var valueline2 = d3.svg.line()
        .y(function(d) { return y(1); })
        .x(function(d) { return x(d.PCA_components); });


    // Get the data
    d3.csv(filename, function(error, data) {
        data.forEach(function(d) {
            d.eigen_value = +d.eigan_values;
            d.PCA_components = +d.PCA_components;
        });

        // Scale the range of the data
        y.domain([0,d3.max(data, function(d) { return d.eigen_value; })+1]);
        x.domain(d3.extent(data, function(d) { return d.PCA_components; }));

        // Add the valueline path.
        svg.append("path")
            .attr("class", "line")
            .attr("d", valueline(data))
            .attr("transform", "translate(40,0)");
            // .attr("data-legend",function(d) { return d.name});

        svg.selectAll("dot")
            .data(data)
            .enter().append("circle")
            .style("fill", function(d,i){ if(d.eigen_value > 1){return "blue";} else {return "green";}})
            .attr("transform", "translate(40,0)")
            .attr("r", 4)
            .attr("cx", function(d) { return x(d.PCA_components); })
            .attr("cy", function(d) { return y(d.eigen_value); });

        svg.append("path")
            .attr("class", "line")
            .attr("transform", "translate(40,0)")
            .attr("d", valueline2(data))
            .attr("style","stroke:red" );
```

Scatterplot_matrix.js

```
function scatterplot_matrix(filename) {
filename = "./data/" + filename;
svg.selectAll("*").remove();

var width = 960,
    size = 400,
    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.category20c();

d3.csv(filename, function(error, data) {
    var domainByTrait = {},
    traits = d3.keys(data[0]).filter(function(d) { return d ;}),
    n = traits.length;

    traits.forEach(function(trait) {
        domainByTrait[trait] = d3.extent(data, function(d) { return d[trait]; });
    });

  xAxis.tickSize(size * n);
  yAxis.tickSize(-size * n);

  var brush = d3.svg.brush()
              .x(x)
              .y(y)
              .on("brushstart", brushstart)
              .on("brush", brushmove)
              .on("brushend", brushend);

  svg.attr("width", size * n + padding)
      .attr("height", size * n + padding)
      .append("g")
      .attr("transform", "translate(" + padding + "," + padding / 2 + ")");
```

elbowplot.js loadings1.js scatter_plot.js scatterplot_matrix.js assignment2.html screeplot.js

```
  // Titles for the diagonal.
  cell.filter(function(d) { return d.i === d.j; }).append("text")
      .attr("x", padding)
      .attr("y", padding)
      .attr("dy", ".71em")
      .text(function(d) { return d.x; });

  cell.call(brush);

  function plot(p) {
    var cell = d3.select(this);

    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);

    cell.append("rect")
        .attr("class", "frame")
        .attr("x", padding / 2)
        .attr("y", padding / 2)
        .attr("width", size - padding)
        .attr("height", size - padding);

    cell.selectAll("circle")
        .data(data)
        .enter().append("circle")
        .attr("cx", function(d) { return x(d[p.x]); })
        .attr("cy", function(d) { return y(d[p.y]); })
        .attr("r", 4)
        .style("fill", function(d,i) { return color(i); });
  }

  var brushCell;

  // Clear the previously-active brush, if any.
  function brushstart(p) {
    if (brushCell !== this) {
      d3.select(brushCell).call(brush.clear());
      x.domain(domainByTrait[p.x]);
      y.domain(domainByTrait[p.y]);
      brushCell = this;
    }
  }

  // Highlight the selected circles.
  function brushmove(p) {
    var e = brush.extent();
    svg.selectAll("circle").classed("hidden", function(d) {
      return e[0][0] > d[p.x] || d[p.x] > e[1][0]
          || e[0][1] > d[p.y] || d[p.y] > e[1][1];
    });
  }

  // If the brush is empty, select all circles.
  function brushend() {
    if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
  }

  function cross(a, b) {
  var c = [], n = a.length, m = b.length, i, j;
  for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y: b[j], j: j});
  return c;
}
```

```
def determine_pca(data_frame,type):
    data_frame = data_frame.as_matrix()
    kArr = list(xrange(17))
    # kArr.pop(0)
    pca = PCA(n_components=2)
    print (pca)
    print "DATA",data_frame
    pca.fit_transform(data_frame)
    print "Eigen vectors ",pca.components_

    print "Eigens ",len(pca.components_)

    print "After performing eigen"

    temp = []
    for i in range(0,17):
        temp.append(pca.components_[0][i] * pca.components_[0][i] + pca.components_[1][i] * pca.components_[1][i
            ])

    print "temp ",temp,len(temp)
    objects = ('stateur','statemb','state','age','tenure','yrdispl','rr','NumericHead','Numericnwhite',
        'Numericnschool12','Numericsex','Numericbluecol','Numericsmsa','Numericmarried', 'Numericdkids',
        'Numericdykids','Numericui')
    screeDataFrame =  pd.DataFrame({'PCA_components' : objects})
    screeDataFrame['eigen_value'] = temp
        # sample.to_csv(file_name, sep=',')
    if type == 1:
        screeDataFrame.to_csv('randomscreeplt.csv', sep=',')
    else:
        screeDataFrame.to_csv('adaptivescreeplt.csv', sep=',')#adaptivescreeplt
    print "variance ",pca.explained_variance_
    return pd.DataFrame(pca.fit_transform(data_frame))
```

```python
def find_MDS(dataframe, type):
    dis_mat = SK_Metrics.pairwise_distances(dataframe, metric = type)
    mds = MDS(n_components=2, dissimilarity='precomputed')
    return pd.DataFrame(mds.fit_transform(dis_mat))

data_directory = "/Users/karanmalhotra/Downloads/"
def create_File(random_sample, adaptive_sample, file_name):
    # print "random ",random_sample
    # print "adaptive ",adaptive_sample
    # random_sample['type'] = 1
    # adaptive_sample['type'] = 2
    random_sample.columns = ["r1","r2"]
    adaptive_sample.columns = ["a1","a2"]

    sample = random_sample.join([adaptive_sample])

    file_name = data_directory + file_name
    sample.to_csv(file_name, sep=',')



def calculate_values(random_sample, adaptive_sample,function,file_name):
    create_File(function(random_sample,1), function(adaptive_sample,2),file_name +".csv")


def Apply_Normalization(dataFrame):
    # dataFrame = dataFrame.apply(zscore)
    # return dataFrame
    min_max_scaler = preprocessing.StandardScaler()
    np_scaled = min_max_scaler.fit_transform(dataFrame)
    df_normalized = pd.DataFrame(np_scaled)
    return df_normalized

def main():
    benefitDF =  pd.read_csv("/Users/karanmalhotra/Downloads/Benefits.csv")
    precision = 3
    benefitDF['rr'] = benefitDF['rr'].round(decimals = precision)
    # print benefitDF['rr']
```

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

Interesting observations about data

The dataset had 18 attributes, out of which only 4 has eigen value 1 or higher. The intrinsic dimensionality of the data was 4 for a dataset of 18 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 could also have used 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