Student Name : Nguyen Xuan Binh
Student Number: 887799

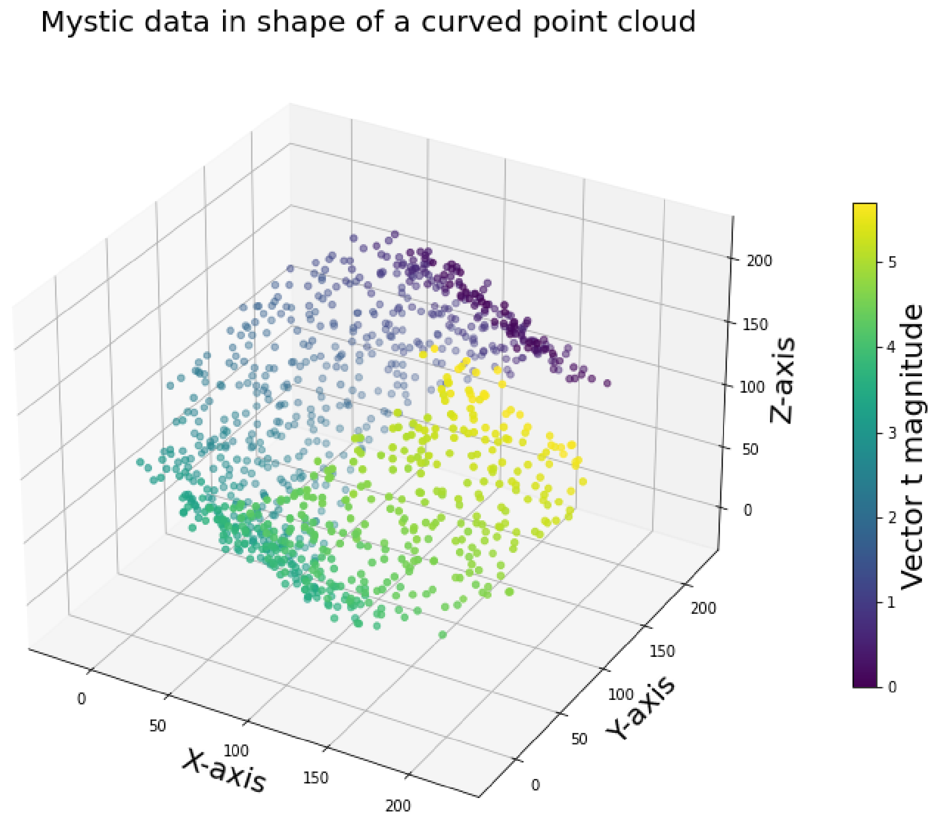
Information Visualization Assignment 3

Exercise 1 (7 points)

Download the dataset mysticdata.csv, which contains n = 1000 data points. The first real-valued column is a vector and the next (2nd–4th) real-valued columns are the data in m = 3 dimensions—the data matrix X. The data presents a curved point cloud in three dimensions. This exercise aims to study the shape of this three-dimensional data set by embedding it into one or two dimensions. The items (b)–(e) below should contain a brief explanation of what you have done.

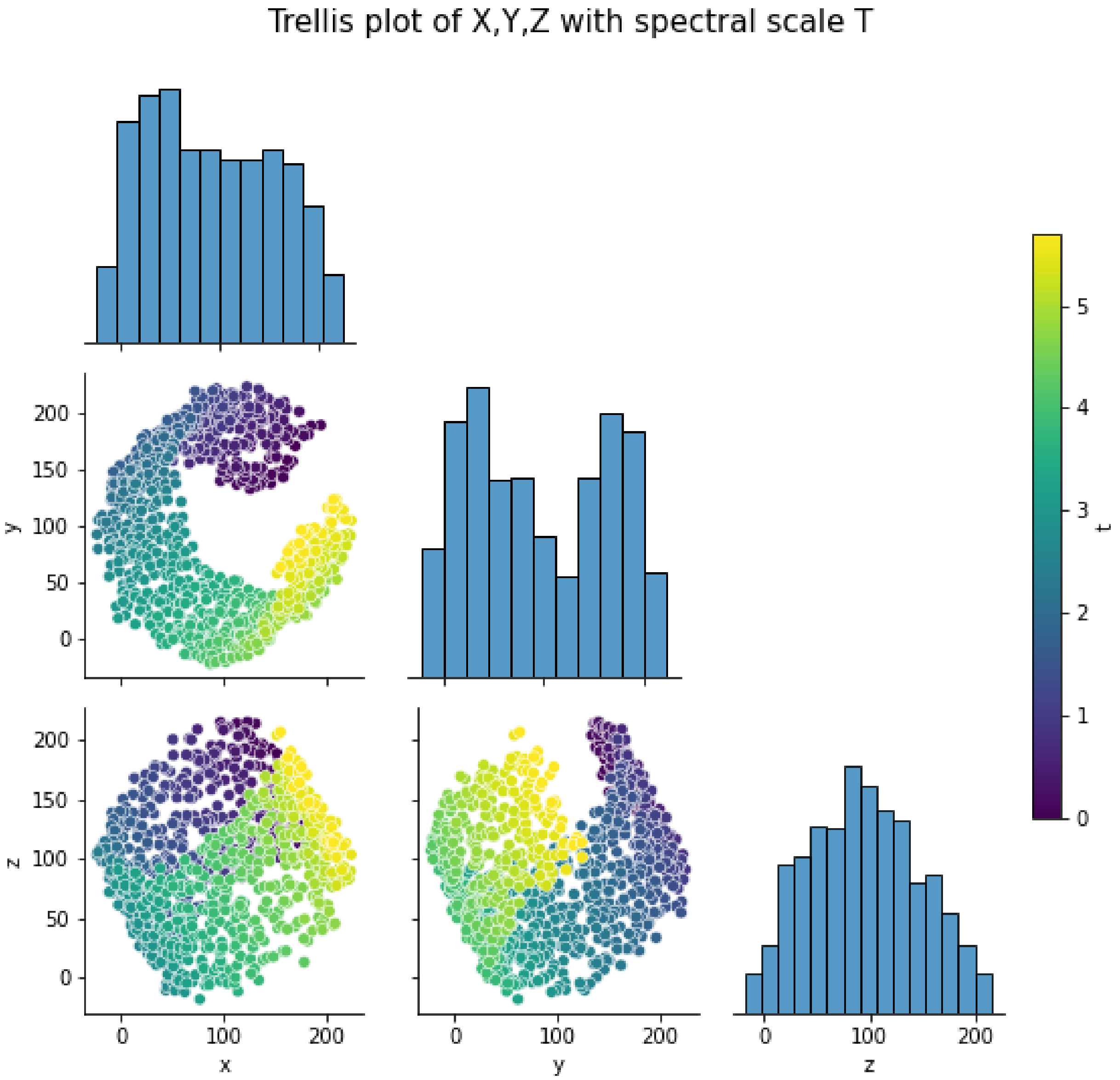
Hints: You can use any software you want to do this and the next exercise. In the lecture slides, you can find links to some examples made with R that you may find helpful. As a measure of the neighborhood, a good definition is that items i and j are neighbours if j is one of the k (k < 10) closest points to i or if i is one of the k closest points of j, but you can also use some other denition of neighbourhood.

First, we can look at the data in 4D to have a general view of the dataset. From this view, we cannot make any conclusion apart from the continuity in the spectral color value of t (MATLAB)



(a) Make a trellis (small multiples, similar to Exercise 4 in Assignment 1) of 2-dimensional scatterplots of the point set in X such that you map the value of vector t to a suitable continuous colour scale (e.g., spectral scale) (done in Python)

This exercise part asks to construct a figure of pairplots for the three columns, x, y, z, with the hue based on the magnitude of the associated t-value. The chosen spectral scale is the viridis scale as this scale has two color channels that avoid the problem of color-blindness (red-green)

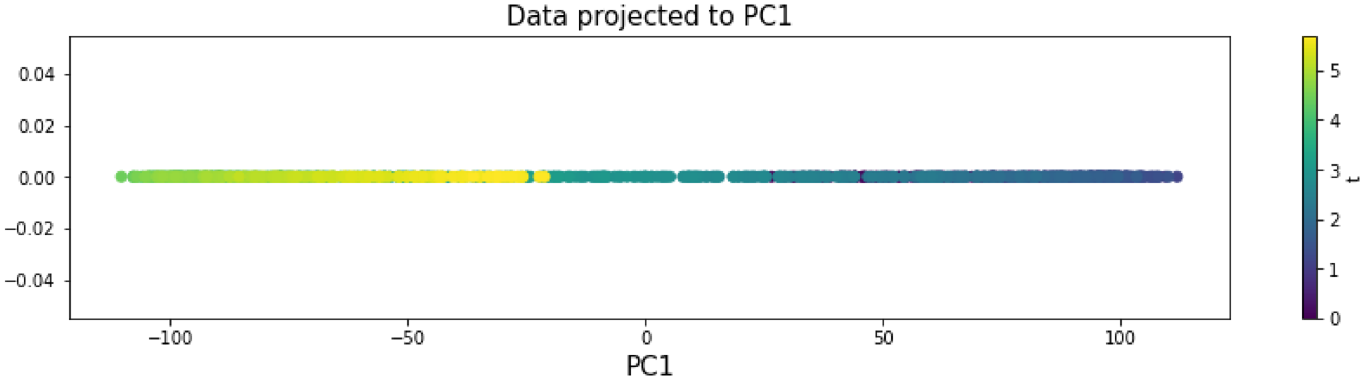


First, we can take a close look at the diagonal of histograms. Column x seems to be uniform, while column y seems to be bimodal, and z seems to be normal. This will help us infer about the shape of the dataset in 3D, as the scatterplots that have y colume resemble the C character, while for x and z, the gap of the C character is not visible. However, we still cannot make any conclusion whether the shape of the dataset is really a gapped cylinder (C character) in 3D.

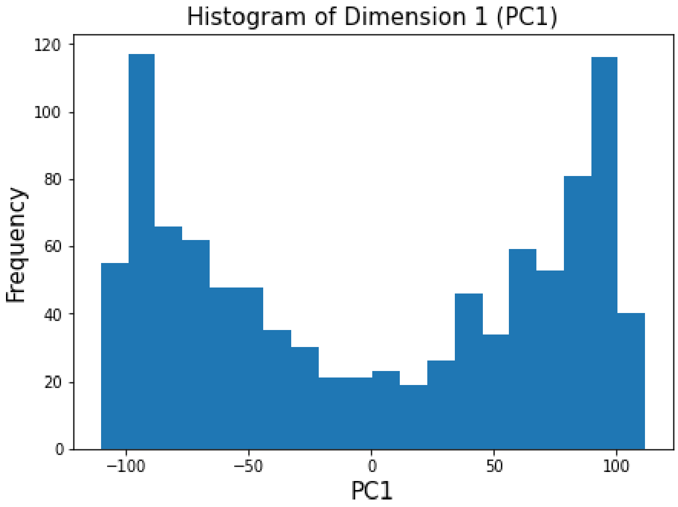
(b) Use pca, project the data to the first principal component, and make a plot of the data into one dimension using the same colour scale as in item (a) above. Also, make a histogram of the one-dimensional embedding. With pca, it is a good idea to center the data first. Why? What would happen if the data would be uncentered when looking for a maximum variance projection? (done in Python)

It is generally common practice to center the data first before applying PCA or standardizing the data if the variance is not the same between the features. This is because PCA is based on the covariance matrix, which is highly sensitive to the scale and center of the variables. The columns x,y,z share the same range, so only centering is enough in this case. Centering the data ensures that the features have zero mean and allows the PCA to capture the actual maximum variance in the data structure. If the data is not centered, the principal components will not reflect the true shape of the data. Specifically, maximum variance projection is influenced by the differences in the features means, which affects the correlation between them.

This is the project data to PC1 with its respective colour scale as in item (a).



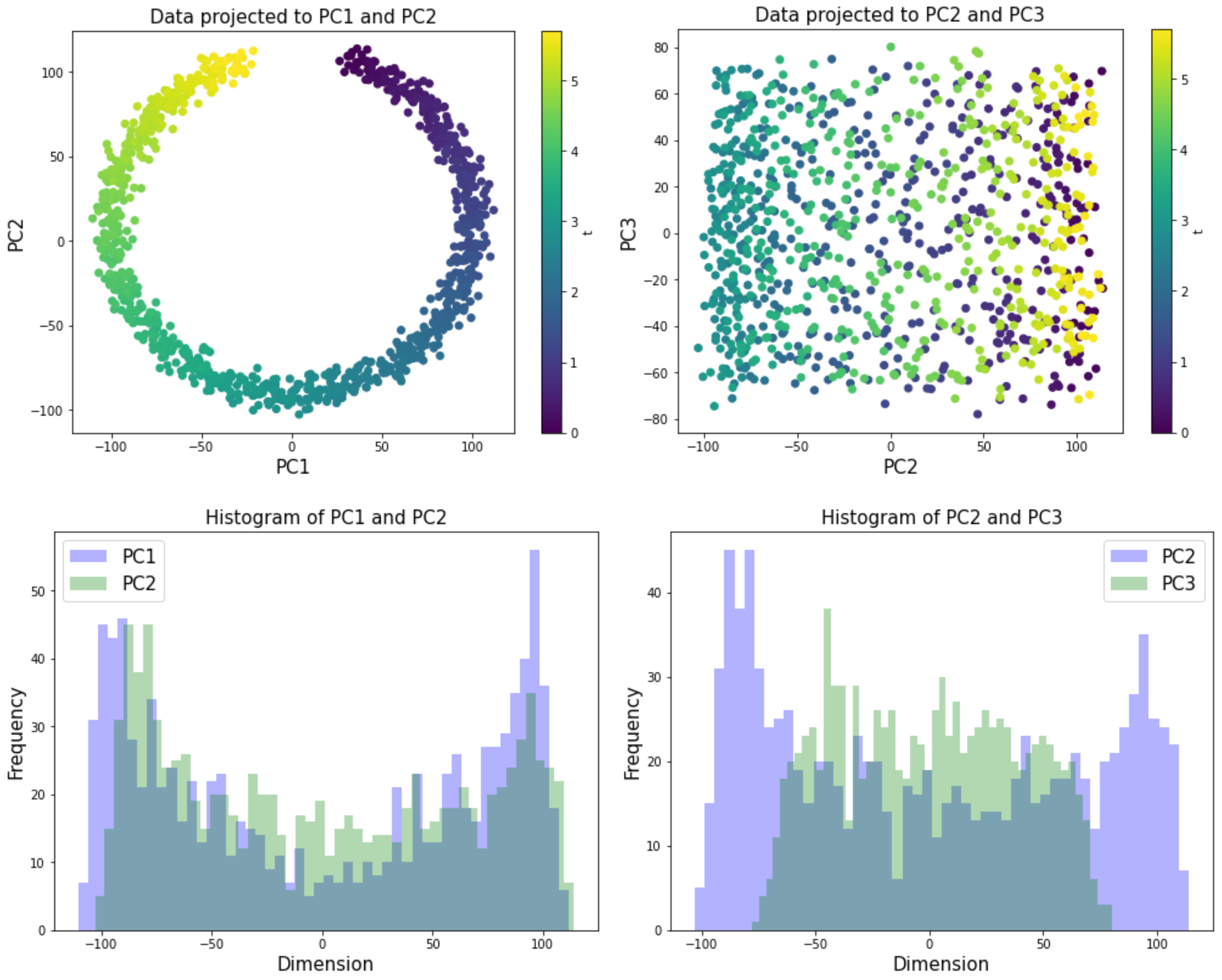
The histogram of the one-dimensional embedding PC1



As we can observe, the bimodal distribution hints that the PC1 reflects the y-axis direction, with few datapoints in the middle and concentrated datapoints at the two ends.

(c) Then make two-dimensional plots of the data projected to the first and second pca components, and to the second and third components. Based on these, what can you tell about the data set's shape? (done in Python)

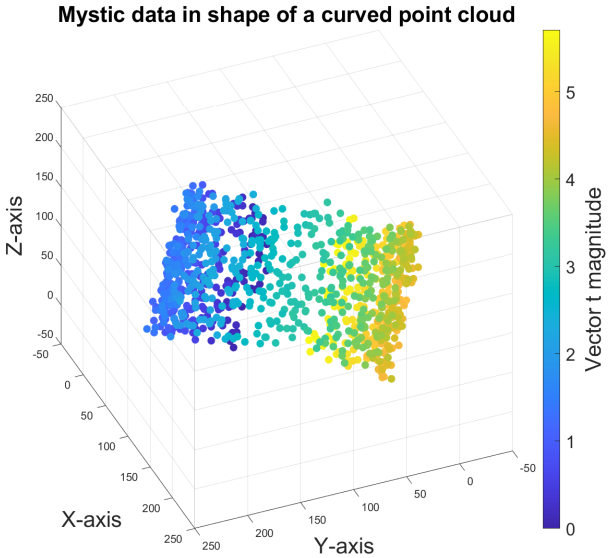
These are the required plots in this exercise



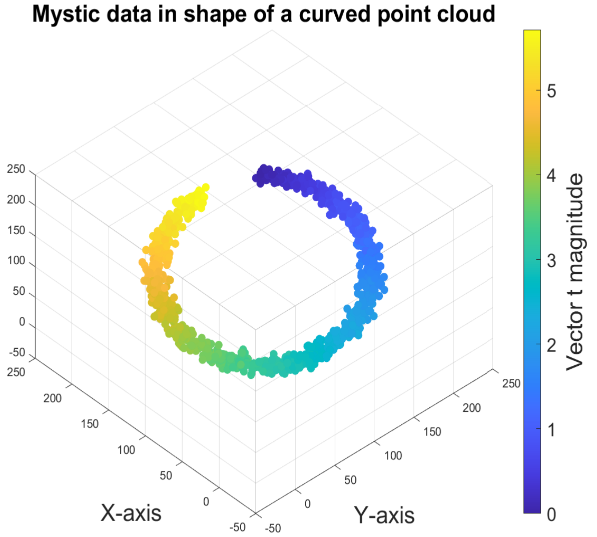
Based on these figures, this is my conclusion about the shape of the dataset in 3D:
- It has a main shape of circle along one dimension (The circle shape in PC1-PC2 component)
- It is a extruded cylinder along that dimension (The overlapping shape in PC2-PC3 component)
- It is an incomplete cylinder with a vertical gap, resulting a C-shape cylinder

This is confirmed in the histogram, as the distribution of PC1 and PC2 are both bimodal. The bimodal distribution hints that data is sparse at the middle, which corresponds to the missing part in the C-shaped cylinder. This gap is highly visible in the projected PC1-PC2 graph

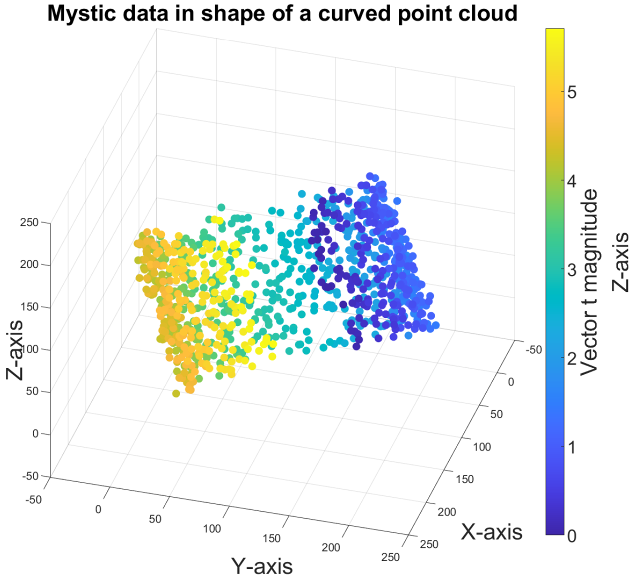
To verify if this is correct, we can plot this in 3D and view from different angles (MATLAB):



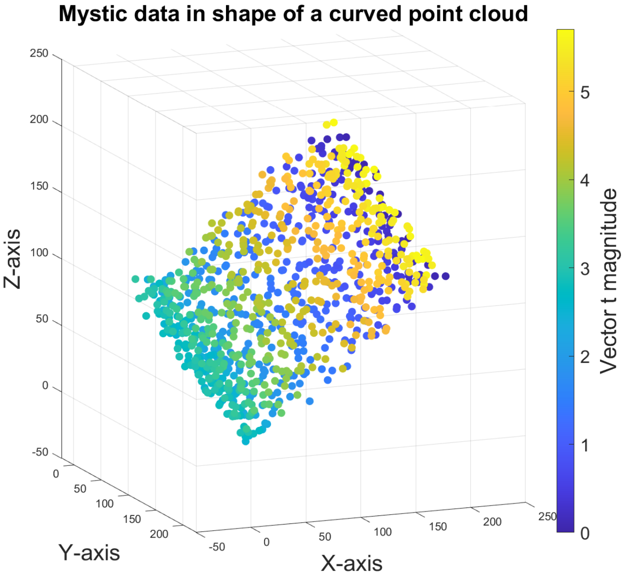
PC1 projection



PC1-PC2 projection



PC1-PC3 projection

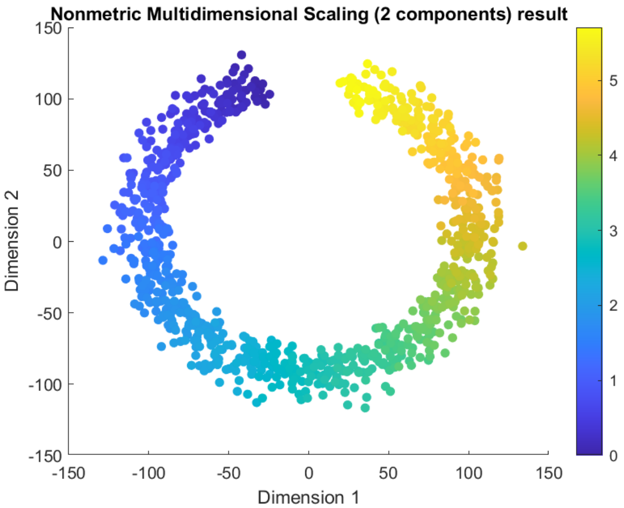


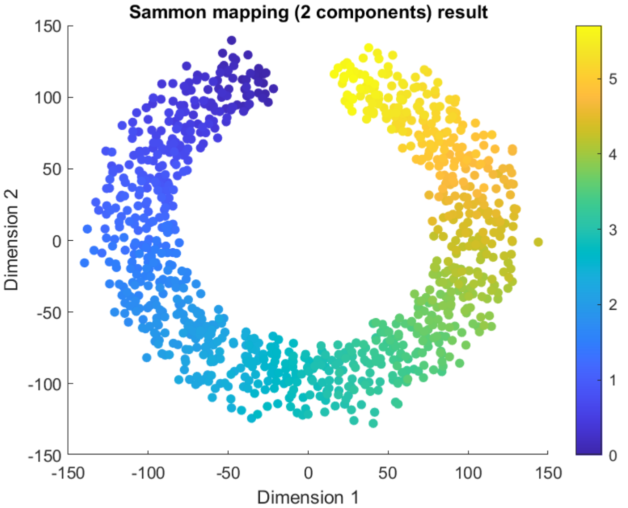
PC2-PC3 projection

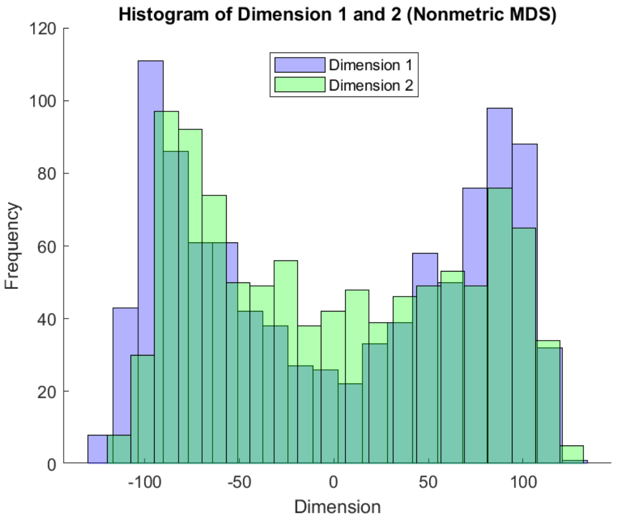
It is true that the shape of the dataset in 3D is a C-shape cylinder ring, with the continuous spectral color range from the increasing values of t from one end of the ring to the other end.

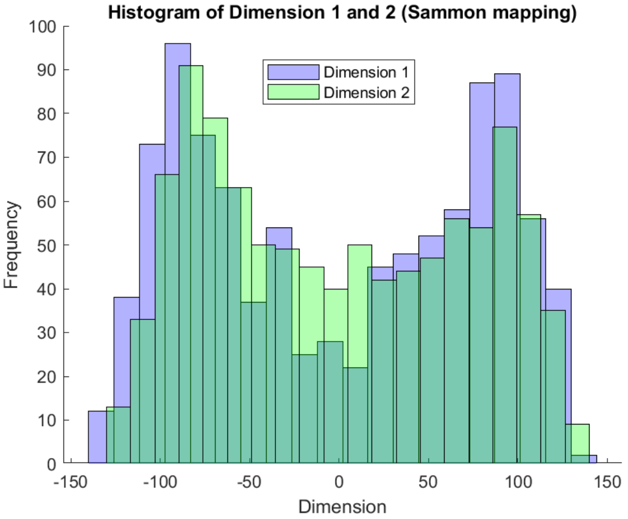
(d) Use nonmetric MDS or Sammon mapping to embed the data into one or two dimensions and plot the data the same way you did in item (b) above (done in MATLAB)

I use both nonmetric MDS and Sammon mapping to reduce 3 dimensions x,y,z down to 2 components. This is the result figures of projected data in the two methods and the histograms.


Nonmetric Multidimensional Scaling (2 components) result


Sammon mapping (2 components) result

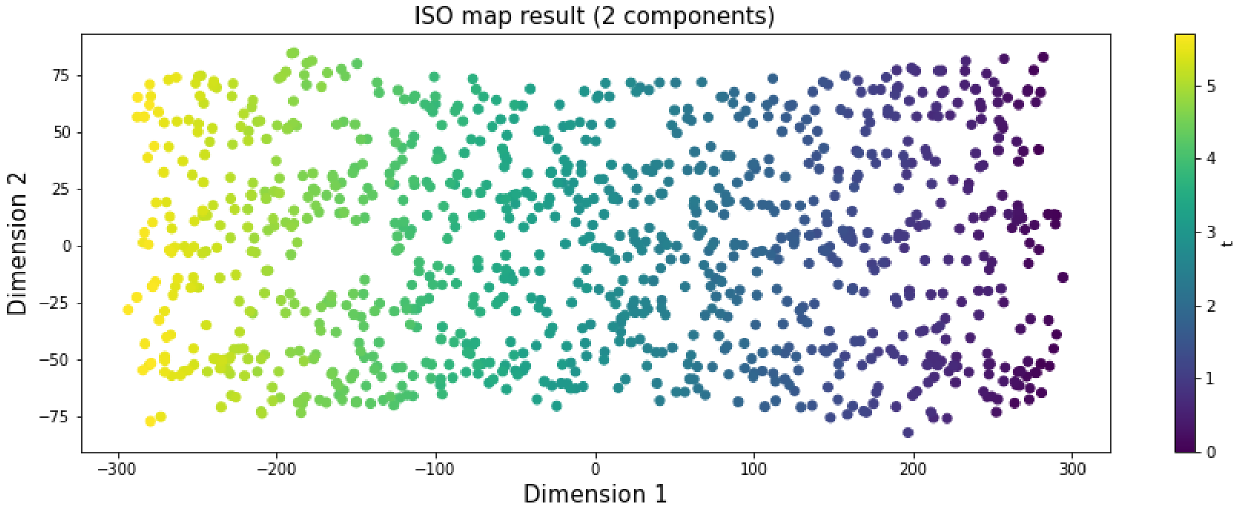

Histogram of Dimension 1 and 2 (Nonmetric MDS)

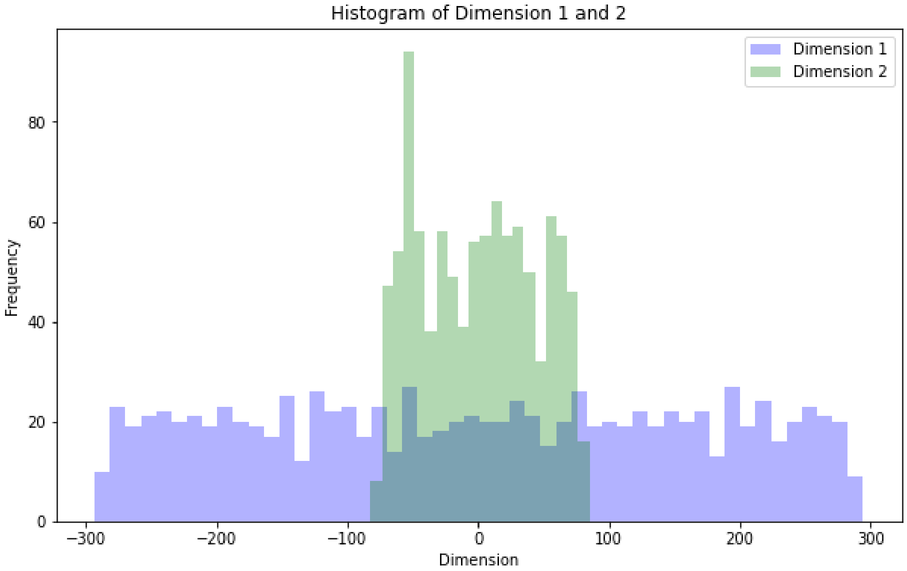

Histogram of Dimension 1 and 2 (Sammon mapping)

The two reduced components correspond to the PC1-PC2 direction completed in part 1c. The two above figures have a C-shape figure and their respective histograms below follow the bimodal distribution, which reflects the gap in the C-shape. In nonmetric MDS, I use the option **mdscale(dissimilarities1, 2,'criterion','stress')**, which is a nonlinear metric. The structure and shape between nonmetric MDS and Sammon are the same, except that Sammon mapping are much more spread out compared to nonmetric MDS.

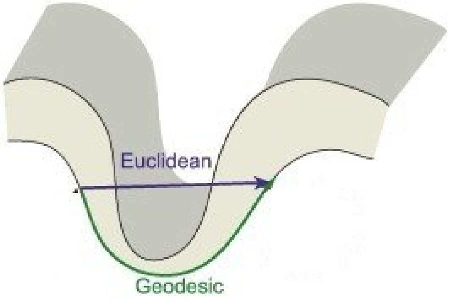
(e) Use Isomap, discussed in the lectures, to embed the data into one or two dimensions and plot the data the same way you did in item (b) above. (done in Python)

This is the ISO map graph and their respective histograms with two and one components. In this exercise, I choose the number of neighboring points as 20 to achieve better smoothness.

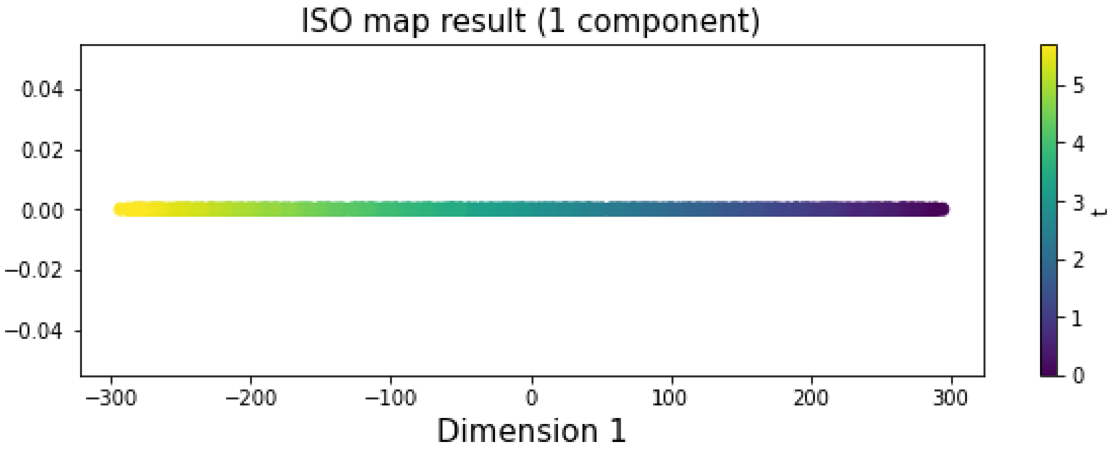


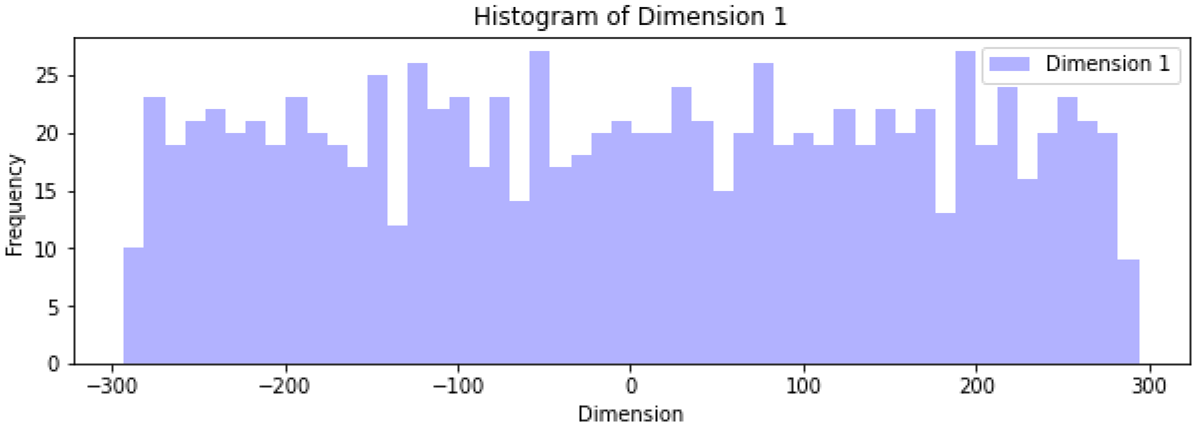


The difference between ISO map and MDS/Sammon is that Euclidean distance is preserved in MDS methods, while the geodesic distance is preserved in ISO map.



In other words, the ISO map tries to capture the true shape of the C-cylinder by traversing along the circle. This is observed in the figure above, where Dimension 1 has a very long range (-300, 300), which basically unwraps the C-cylinder into a rectangle shape. In MDS, the two points on two ends of the C-cilinder will be closest, while in geodesic distance, they are the furthest since the shortest path is to traverse along the shape of the cylinder. Now we can project the dataset onto 1 component using ISO map





The figure shows that the transition of values of t truely reflects the spectral scale along the circle around the C-cylinder with a smooth spectral scale. As a result, ISO map is capable of explaining complex shape much better than MDS methods, such as the Swiss role and C-cylinder dataset. The histogram also reveals the true uniform distribution of t-values along the C-cylinder, which means ISO map can capture the pattern of vector t magnitude.

Exercise 2 (4 points)

Download from MyCourses the dataset population_data.csv which contains statistics of population age structure in Finnish municipalities. The data is organized in different age groups with the following columns:

1. Name of the municipality 2. Total population in that area

3. People younger than 4 years 4. People of age 5–9 years

…

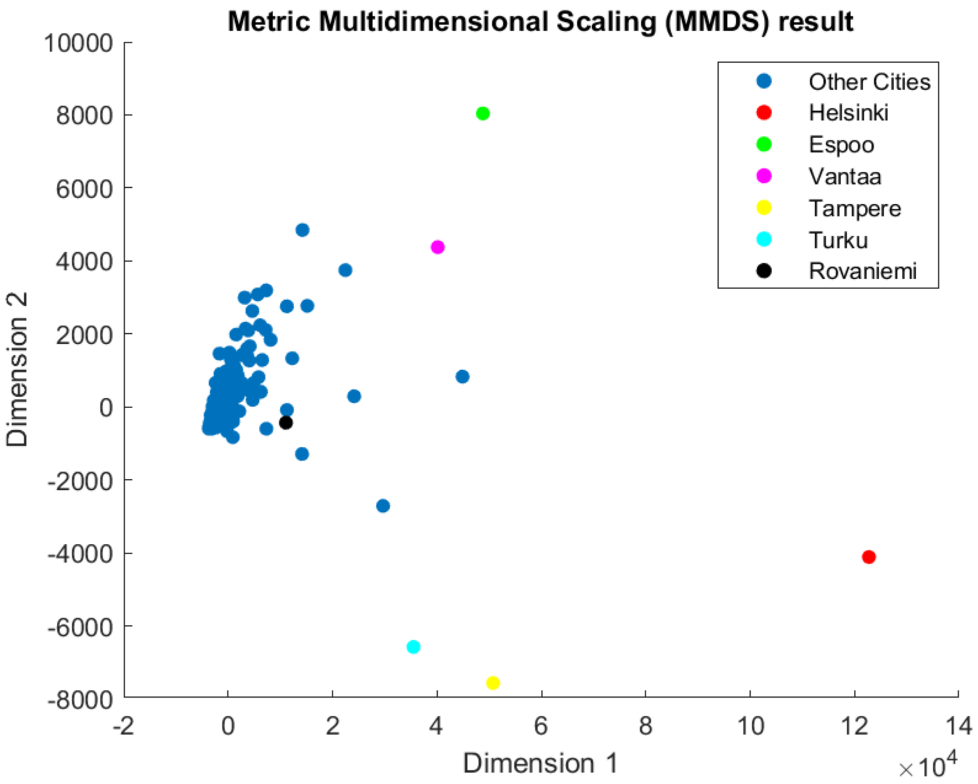
20. People of age 85–89 years 21. People 90 years or older

Compute and plot Metric mds (mmds) and Sammon mapping. Annotate some selected (or all) places, for example, main cities, provinces, places where you have been/born, etc. Compute Shepard plot (a scatter plot of output distances as a function of input distances) and compare the plots of mmds and Sammon mapping. Which method predicts which distances better? The easiest way to do this exercise is to use Matlab. This exercise is done in MATLAB. The following commands are useful:

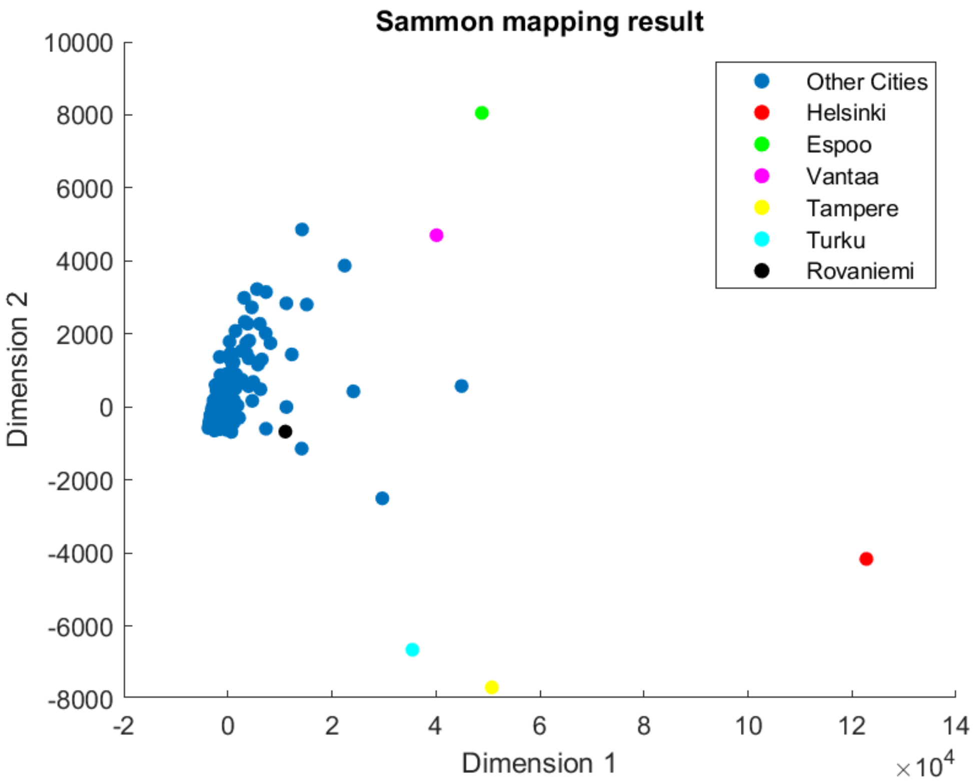
›mdscale: computes various projections. Setting the parameter Criterion to “metricstress” produces mmds. Setting the parameter Criterion to “sammon” produces Sammon mapping.

›pdist: computes Euclidean distance between points.

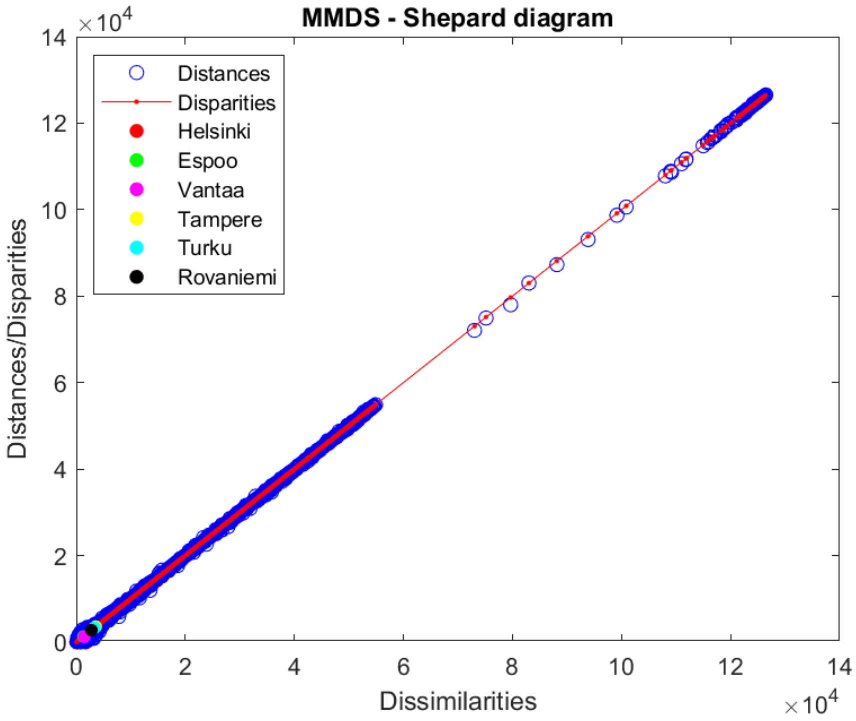
In this exercise, I plotted the MDS and Sammon mapping and their respective Shepard plot in all cities across Finland. Additionally, I also mark a few notable cities, which are Helsinki, Espoo, Vantaa, Tampere, Turku and Rovaniemi to show that they are different from most Finnish cities. First, this is the Metric MDS graph for the age group difference based on Euclidean distance.

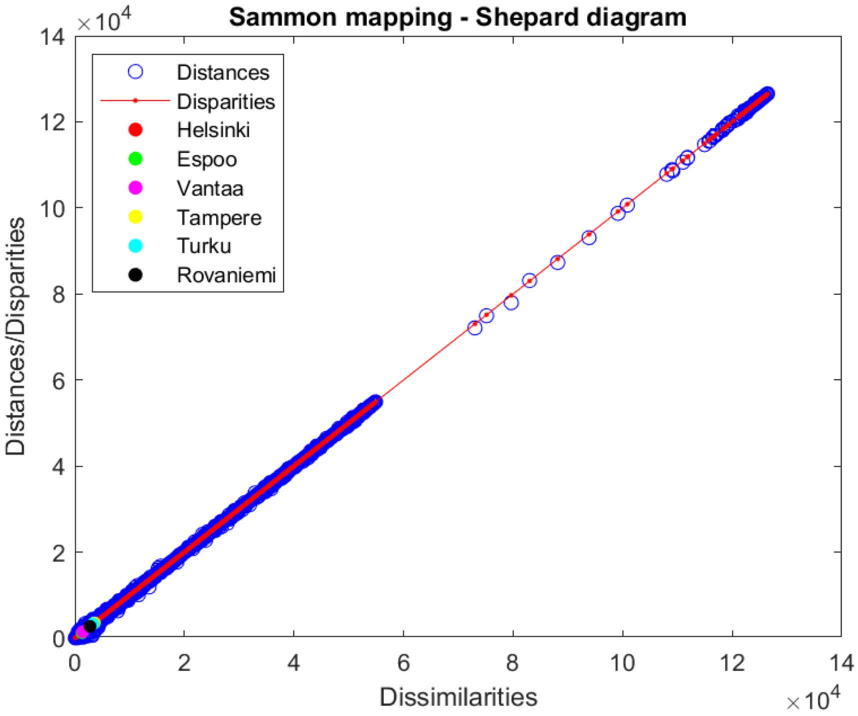


From the figure, we can see that most Finnish cities have the same age demographic structure with only a few exceptions. For example, Rovaniemi is quite similar to most Finnish cities, but it is not the same for the rest 5 cities, which happens to be the top 5-6 most populated cities in Finland. Because cities Helsinki-Turku are the central cities in Finland, they have a different age demography, partly due to job opportunities/immigration that increases the ratio of young people. The most notable is Helsinki, which is a far outlier to any other Finnish cities as it is the capital city, which has attracted young people/migrant workers much more than other cities. This factor greatly contributes to the age difference of Helsinki compared to other cities in Finland. Now I proceeded to plot the Sammon mapping graph, which is indistinguishable from MMDS.



Based on these graphs alone, it is hard to say which method performs better. Therefore, I proceed to implement the Shepard plot. By definition, the Shepard plot is a plot of two measurements of the distances between objects. One measurement is the true distance, and the other measurement is the apparent distance in some representation of the objects. For example, the apparent distance between objects in a photograph (two dimensions) and the real three-dimensional distance. The diagram is used in multidimensional scaling to assess the extent of any distortion. Zero distortion would correspond to a set of collinear points. In the figures below, an ideal setting (perfect matching - all distances are preserved) means that the red line is a perfect straight line with slope equal to 1





From the Shepard plots, we can see that both methods produce almost perfect straight red line, which is the disparities between the original and transformed data. This means they preserve the relative distance very well between each pair of cities. We can also observe that the most populated cities are also concentrated in a cluster, which means they share many similarities in age groups compared to other Finnish cities. However, we are still unsure which method performs better as both Shepard plots look identical, so numerical results are required.

These metrics can be obtained in MATLAB, where stress is produced by the algorithm, and the Spearman's rank correlation coefficient means how much the ranks between each city are preserved. On the other hand, the average relative error is the average difference between the original distances and the distances in the reduced dimensional space, normalized by the original distances. Its formula is:

average relative error = sum(abs(dissimilarities - distances)) / sum(dissimilarities);

	Metric MDS (MMDS)	**Sammon mapping**
Stress achieved	0.007283	0.000768
Spearman's rank correlation coefficient	0.9969	0.9987
Average relative error	0.00802	0.00714

Sammon mapping achieved lower stress value than MMDS given the same number of iterations => Sammon mapping is better based on stress achieved

Regarding the rank correlation coefficient, Sammon mapping aims to preserve the ranks, while MMDS aims to preserve the relative Euclidean distance between each city, so Sammon mapping is better based on rank preservation

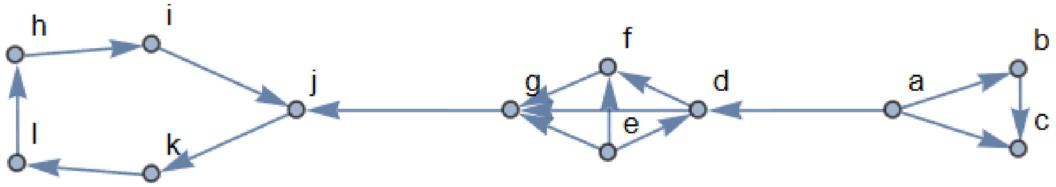
Regarding the average relative error, MMDS is better than Sammon mapping with the above mentioned reason. So in this case, which method performs better depends on whether we want to preserve the rank (Sammon) or preserve the distance (MMDS). However, based on the stress value, I may conclude that **Sammon mapping performs marginally, but not decisively, better than MMDS.**

Exercise 3 (3 points)

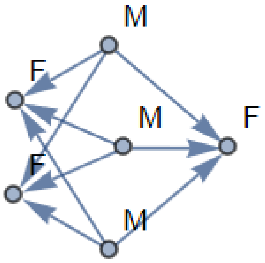
From Mycourses, download the dataset network_data.tgf, which contains a network defined as an adjacency list (explained below). Visualize it using the principles introduced in the last lecture (and the general Tufte's principles taught earlier). Explain why your visualization is appropriate for this network and how you produced it. Also, visually indicate each node's given attribute labels and try to make different network substructures visible.

You may use any software (e.g., yEd, PowerPoint, Illustrator, etc.) or even hand drawing to develop and present your solution. The tgf (Trivial Graph Format) representation used in network data.tgf is a text file containing first a list of nodes (on each line a node identifier followed by possible attributes), and then a list of edges (pairs of node identifiers), see https://en.wikipedia.org/wiki/Trivial_Graph_Format

First, I use Mathematica to plot the graph in the tgf format. We can already see some patterns here in the graph, but there are few details that we must do, such as removing the directed edges, since tgf format does not indicate it is a directed graph.

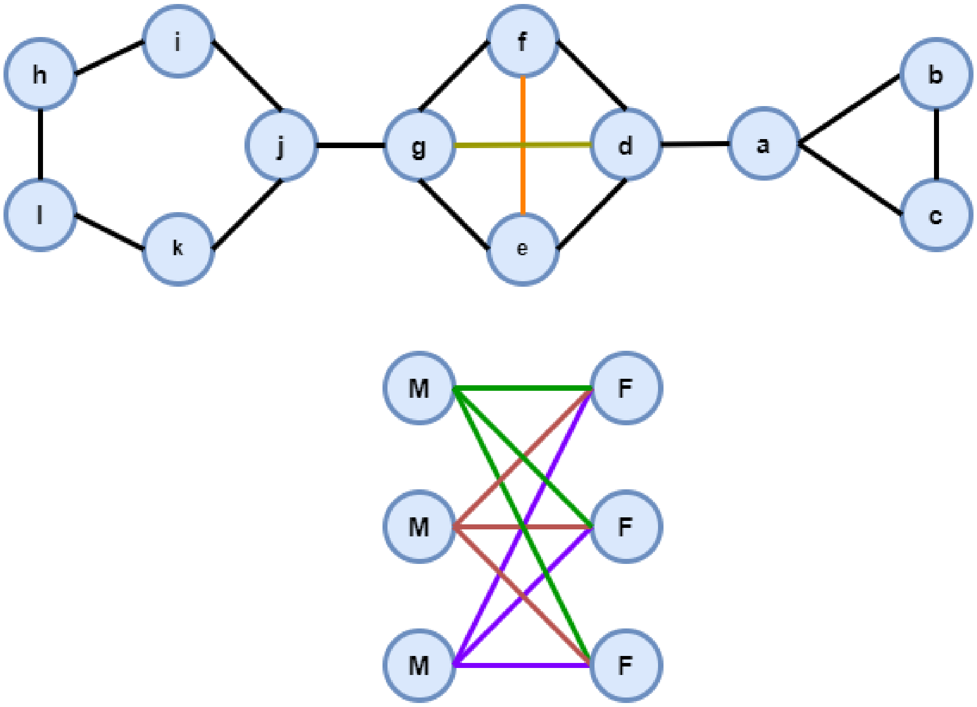


Out[22]=



The above graph produced by Mathematica also does not obey the visualization rules, so I proceed to use draw.io (https://app.diagrams.net) online drawing tool to rearrange the graphs.

This is the final graph I plotted for this exercise



First, I need to determine the general layout, as visualization greatly determines how a graph is interpreted. I choose the spring (force-directed) layout, which is a layout simulated by spring force attached to every edge that minimizes the total tension of the springs. This layout tends to make distances equal and clustered nodes together, which focuses on separating main cluster types into subgraphs connected by the spring edges. The graph above clearly has 4 subgraphs, the pentagon h-i-j-k-l, the fully connected diamond f-g-e-d, the triangle a-b-c and the disjoint bipartite graph M-F. Therfore, the spring layout is the most appropriate option for this graph.

Next, I need to make sure the graph obeys the criteria for graph visualization. The graph above in fact has obeyed most criteria as follows

- Vertices, edges, and labels are clearly distinguishable and evenly distributed.
- There is adequate space between vertices.
- Minimize the edge bending ratio. The edge bending ratio is 0 as all the edges in the graph above does not have any bends.
- Minimize the edge lengths, which helps readers detecting the relations among different nodes faster. Each edge in the graph is very short
- Edge-crossings should be minimized. There is only one edge crossing in the diamond subgraph, and the edges in the bipartite graph have been colored differently to deliver the semantic information (each M is associated with all F and each F is associated with all M, dividing them into two disjoint sets M and F)
- The data is inherently structured, so the vertices are distributed the nodes into subgraphs. This increases the understandability of the underlying graph.
- Depict symmetric subgraphs consistently in the form of polygons.
- Aesthetical appearance that follows the Tufte's principles: maximum data-ink ratio, no chartjunk, small multiples and good aesthetics (correct font size, colors are clear, etc)