# Browser-Based Visualization of Networks, trees multivariate data
# An Insight Report

CS732: Data Visualization

Tanishq Jaswani
*International Institute of Information Technology, Bangalore*
IMT2018077
Tanishq.Jaswani@iiitb.ac.in

***Abstract*—CDC's Division of Population Health provides cross-cutting set of 124 indicators that were developed by consensus and that allows states and territories and large metropolitan areas to uniformly define, collect, and report chronic disease data that are important to public health practice and available for states, territories and large metropolitan areas. In addition to providing access to state-specific indicator data, the CDI web site serves as a gateway to additional information and data resources.**

***Index Terms*—Axis swapping, brushing, mapping**

## I. Introduction

We are given tabular dataset which is not so large and hence we will use complete dataset to test some of our hypothesis and will make inferences from same. The data is mainly visualized using plotly.express and pandas library has been used for data manipulation.

We have to generate 4 visualizations, which include:

1) Node-link diagrams with two different layouts, namely, force-directed, circular, including node-labels

2) Matrix visualization of the node-link diagram generated in 1, including node-labels

3) Treemap

4) Parallel coordinates plot with axis swapping and brushing.

## II. Problem Statement

We have been given 4 tasks to accomplish. Following are the them:

1) Pick at least one network and one multivariate dataset from the aforementioned list.

Network data set we chose based on the bio-diseasome and the multivariate data is Sickle-cell-disease. We found these 2 datasets short, precise and produced beautiful visualization with good inferences.

2) Decide how you would like to use the dataset for visualization – directly or after remodeling.

The dataset given to us was already short and did not have more columns in the first place. SO remodelling the dataset did not make much sense so we used it directly as AH Sickle Cell Disease Provisional Death Counts 2019-2021.csv and bio-diseasome.mtx.

3) Prepare datasets for generating visualizations in the aforementioned list, such that the prepared datasets include at least a network, a tree, and a table.

4) Implement the visualizations given in the aforementioned list.

The network dataset webpages in the network repository contain node-link diagram visualizations. These are used as references for the visualizations you generate.

The network datasets are labeled using numerical indices. Hence, these indices may be used as-is for the visualization tasks.

The tabular data for chronic diseases contain several irrelevant columns. We pick columns based on the semantics of the variable, and known data type (categorical or numerical), as not all the variables are needed to be plotted.

These datasets are considered small datasets. Hence, we have used all rows in the multivariate datasets, and all nodes and edges in the network datasets for visualizations, unless any of this is spurious data. This implies that subsampling these datasets for visualizations is not needed and have not used also.

## III. Visualizations

### A. Part-1

For our part-1 we have to generate Node-link diagrams with 2 different layouts, namely, force directed and circular including node labels.

After importing the data through file path we use from numpy matrix to represent our graph and store it in random variable G. We use coolwarm to visualize our data as default as it presents in a neat way. We use spring layout for the force directed graph visualization and circular layout for circular.

Force directed graph is used to visualize the connections between objects in a network. By grouping the objects connected to each other in a natural way, a Force-Directed Graph is visually interesting and also makes it possible to discover subtle relationships between groups.They are also sometimes

referred to as ”spring-embedder” algorithms or ”energy-based placement” algorithms. This algorithm arranges the graphs in a natural and aesthetically pleasing way, usually resulting in a symmetrical and clustered structure. The distribution of the different ”nodes” is harmonious and well balanced.

The algorithm is based on a physical model. It calculates the layout of the graph based solely on the information contained in the graph structure rather than relying on domain-specific knowledge.

The nodes are represented by points that repel each other like magnets. Borders connect these points by simulating a spring force to attract adjacent nodes. The model determines the forces acting on the nodes iteratively and moves them in order to come as close as possible to an equilibrium where the position of the nodes remains stable.

The great advantage of this type of Force-Directed graph plotting algorithm is the simplicity of its implementation. However, it works best for graphs where the number of lines is similar to the number of points. Denser graphs with many lines or unstructured graphs are less suitable.

Talking about circular layouts. he algorithm partitions the nodes of the graph based on specific criteria, e.g., connectivity and by taking into account the desired layout style, e.g., single or multiple circles. Each such partition constitutes a single node that only has edges to other related partitions. Thus, the resulting graph obtains a tree-like structure (the so-called underlying tree). The algorithm arranges each partition as a circle. Depending on the desired style, each circle has nodes either on the boundary or in the interior. A specialized approach for tree structures calculates the arrangement of the underlying tree. The algorithm moves the partitions to their final location.

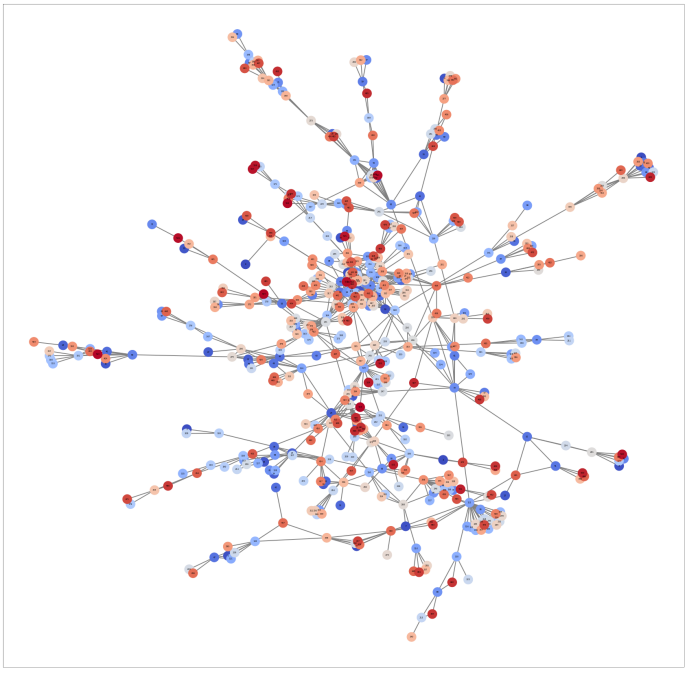

Fig. 1. Force-directed

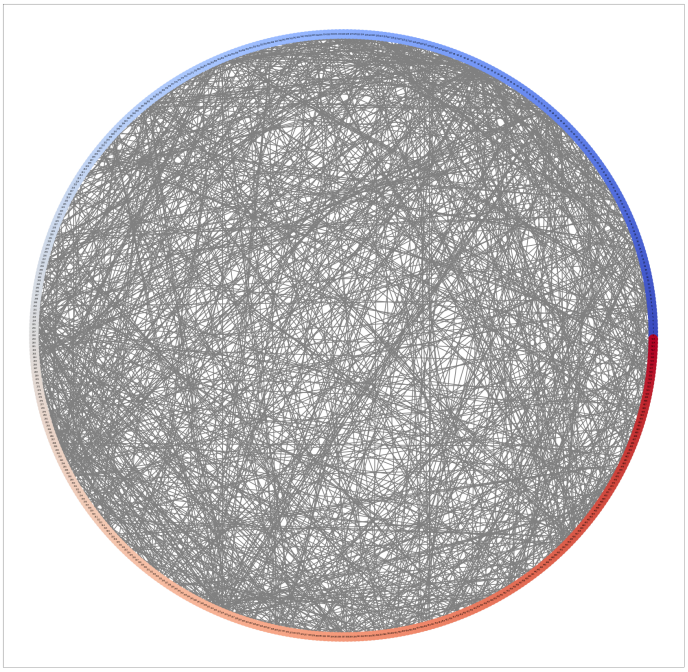

Fig. 2. Circular

### B. *Part-2*

Part-2 of our insight report solves the problem of matrix visualization of the node-link diagram generated in 1, including node-labels.

After generating the graph in the first part we use it to convert to numpy matrix. Then we use heatmap to display our visualization and it looks like the following.

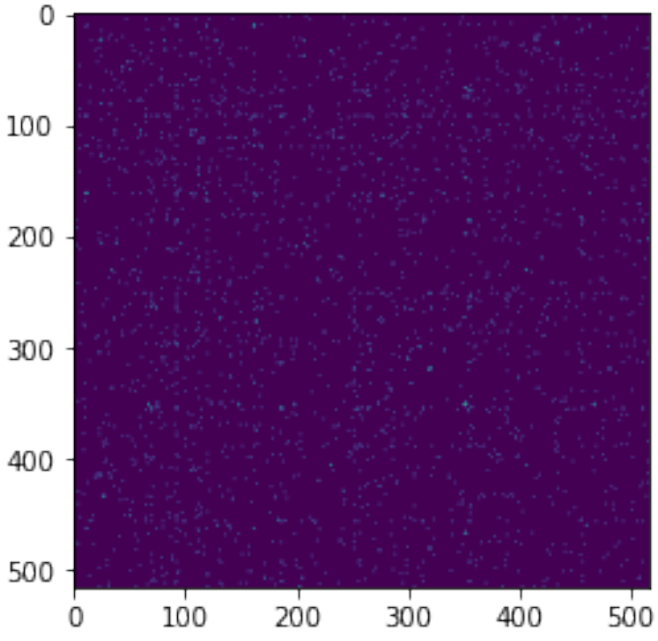


Fig. 3. Part-2

### C. Part-3

We made treemap for visualizing 3 of SCD such as SCD underlying, SCD COVID-19 as well as SCD Multi. We then output it to .html files.

Treemaps are ideal for displaying large amounts of hierarchically structured (tree-structured) data. The space in the visualization is split up into rectangles that are sized and ordered by a quantitative variable.

The levels in the hierarchy of the treemap are visualized as rectangles containing other rectangles. Each set of rectangles on the same level in the hierarchy represents a column or an expression in a data table. Each individual rectangle on a level in the hierarchy represents a category in a column.

The rectangles in the treemap range in size from the top left corner of the visualization to the bottom right corner, with the largest rectangle positioned in the top left corner and the smallest rectangle in the bottom right corner. For hierarchies, that is, when the rectangles are nested, the same ordering of the rectangles is repeated for each rectangle in the treemap. This means that the size, and thereby also position, of a rectangle that contains other rectangles is decided by the sum of the areas of the contained rectangles.

For example describing SCD Underlying tree map, we see it segmenting into first Date of Death Year i.e. 2019, 2020, 2021. We further segment it into quarter naming 1,2,3,4. Then comes Race or Hispanic origin and finishing with Age group.

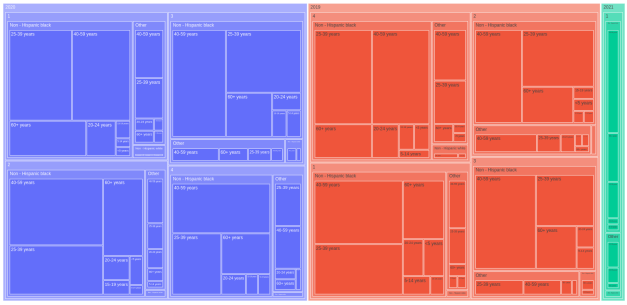

Fig. 4. SCD Underlying

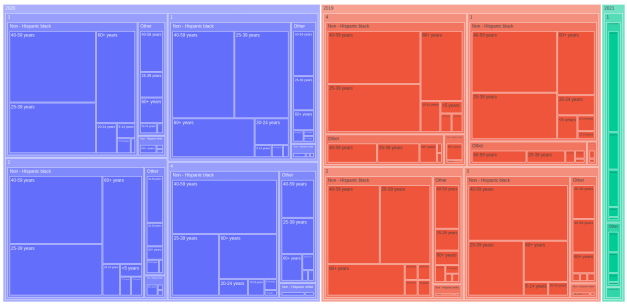

Fig. 5. SCD Multi

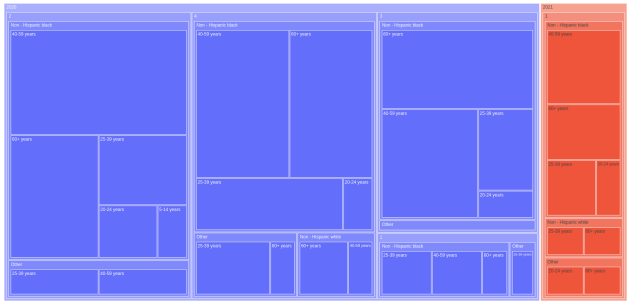

Fig. 6. SCD and COVID-19

### D. Part-4

We made Parallel coordinates plot with axis swapping and brushing for visualizing 3 of SCD such as SCD underlying, SCD COVID-19 as well as SCD Multi. We then output it to .html files.

We started with making a data frame by importing from file by path. To generate the figure we use parallel coordinates. In a parallel coordinates plot, each row of the pandas DataFrame is represented by a polyline mark which traverses a set of parallel axes, one for each of the dimensions.

From the following figures we inference that at a particular date of death year and at a particular quarter how many counts we have of SCD Underlying, Multi and COVID-19 respectively.

A parallel coordinate plot maps each row in the data table as a line, or profile. Each attribute of a row is represented by a point on the line. This makes parallel coordinate plots similar in appearance to line charts, but the way data is translated into a plot is substantially different. The values in a parallel coordinate plot are always normalized. This means that for each point along the X-axis, the lowest value in the corresponding column is set to 0 and the highest value in that column is set to 100 along the Y-axis. The scale of the various columns is totally separate, so do not compare the height of the curve in one column to the height of the curve in another columns.

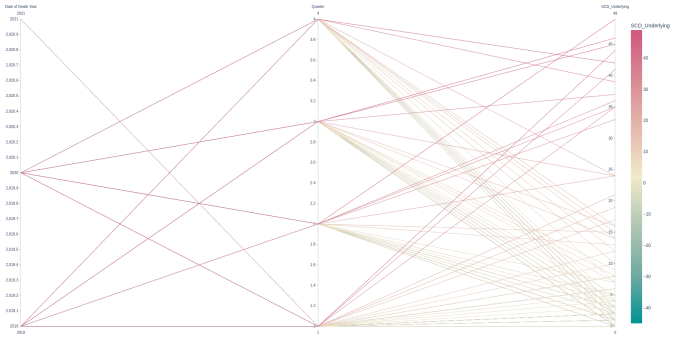

Fig. 7. Parallel - SCD Underlying

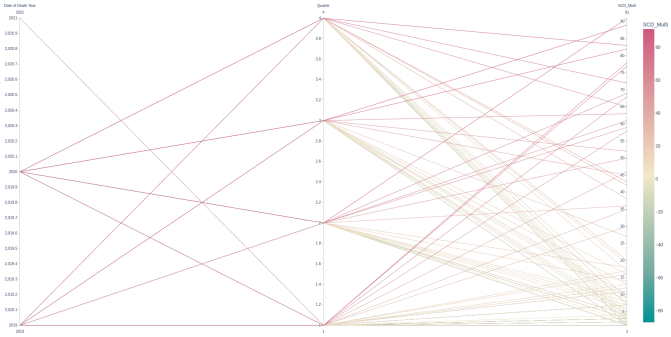


Fig. 8. Parallel - SCD Multi

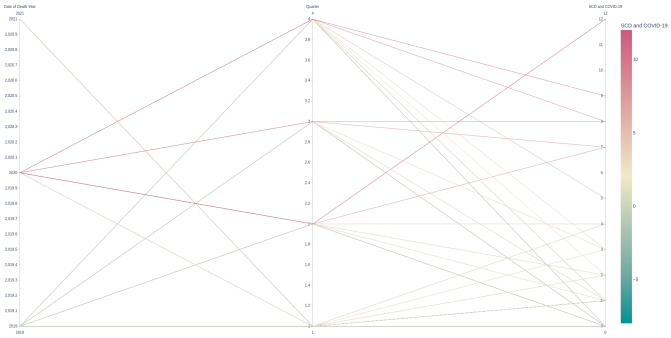


Fig. 9. Parallel - SCD and COVID-19

## IV. Conclusion

In this report we have successfully generated some beautifully visualized Treemaps, Parallel coordinates and circular graph plots. Unlike the other assignments dealing with large datasets we had a small dataset this time which helped us to visualize better and infer alot of information. Working on this datathon made me realise the importance of data visualization. During these times of pandemic, it is very crucial to effectively visualise data so that we can make meaningful inferences from them and take action accordingly. Overall it was a fun activity to do and complete the task within the deadline.

## References

[1] Plotly — Treemaps
[2] Plotly — Parallel Coordinates
[3] Plotly — Force Directed Visualization
[4] Matplot Lib
[5] Plotly — Circular graph plot