Data Visualisation - 3

Agam Kashyap
IMT2018004
International Institute of Information Technology, Bangalore
agam.kashyap@iiitb.ac.in

***Abstract*—A detailed report on visualisations for network data -force-directed graph, circular graph, matrix- and visualisations for multivariate data -treemaps and parallel coordinates plots.**

***Index Terms*—Network, Force-Directed, Circular, Matrix, Treemap, Cascade, Parallel-Coordinates, Plotly, D3**

I. Introduction

This report corresponds to the Data Visualisation Assignment 3. The goal of this report is to explain the methods used to generate network-based visualisations - force-directed graph and circular graph and visual representations for multivariate data- parallel coordinate plot and treemaps. Furthermore, it explains the conclusions that can be made from the visualisations generated.

II. Data

The two datasets that I have chosen to explore are - biodiseasome [1] and Sickle Cell Disease Provisional Death Counts. Biodiseasome dataset consists of data that shows the shared genes between diseases. This is useful in understanding the genetic origins of the diseases. The Sickle Cell disease provisional death counts provides us with data about the death counts of sickle cell disease and coronavirus disease 2019 (COVID-19), by quarter, age, and race or Hispanic origin from 2019 through Quarter 1, 2021.

The Biodiseasome data consists of 516 nodes with a total of 1188 edges between these nodes. The Sickle Cell Disease Provisional Deaths dataset consists of 8 different categories of data namely - Date of Analysis, Death Year, Quater of the Year, Race or Hispanic Origin, Age group and the following numerical categories : Deaths with sickle disease as the underlying cause, deaths with sickle disease as underlying or contributing cause and deaths with sickle disease and covid-19. It has 129 such records in total.

III. Network Data Based Visualisations

A. Force - Directed Graph

For generating the force directed graph I used the d3-force module. There are four basic steps that are occuring to result in the final visualisation : charges on each of the nodes which acts as an attracting or repelling force depending on the charge, an attractive gravity like force component in X and Y which pulls all the nodes towards the specified point, a collision prevention system which prevents overlap of the nodes and weight calculation for each of the node.

As mentioned before each of these nodes denotes some disease and the connection between two nodes shows us the shared genes between the two. Since the data did not come with any specific information about the groups or the disease to which a node corresponds to, I decided to differentiate the nodes based on the number of connections. This is helpful in realising a disease-node which has a higher number of shared genes with others and furthermore as recognising the cluster centers.

Furthermore, in the algorithm the number of iterations plays an important role. With the increased size of the nodes it is important to ensure that there is no overlap of nodes in the generated visualisation. The increased number of iterations ensure better convergence to a solution for the integration involved in the algorithm. In the Fig 7 the visualisation generated have the force along x and y axis activated and hence results in a tightly packed network. Increasing the iterations of the collision system ensures that the smaller nodes do not over lap the bigger nodes. Fig 8 shows the graph without any gravity. In this visualisation one can clearly visualise the different clusters.

B. Circular Graph

The goal of this graph is to place the nodes along the circumference of a circle and connect all the nodes which have an edge. Fig 1 shows the visualisation. You can see the tooltip shows the current node and all the nodes that it is connected to. Furthermore, there is the zoom functionality which can be used to zoom into a specific region and easily access each of the nodes. Currently the ordering is based on just the node id, but a more complex ordering can be according to the clusters that the nodes belong to(implementation of the same was a complex task). For making this visualisation I used the Plotly library of Python and the scatter plot. An interesting thing to note is the way in which the data for the edges is passed. The Scatter function takes in the X and Y values for the edges and draws a line between every two consecutive points. Now, an edge is denoted by say `x1, y1, x2, y2` then we must add (x1,x2,None) to X values and (y1, y2, None) to the Y values. The ”None” is important since it prevents drawing a wrong edge, i.e. connecting (x2,y2) with the next value in the X and Y arrays where no edge exists.

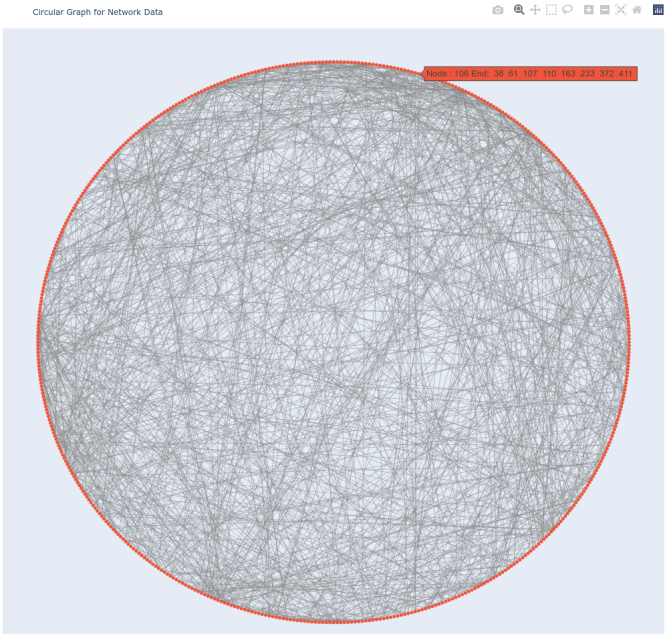

Fig. 1. Circular Graph generated for the network data

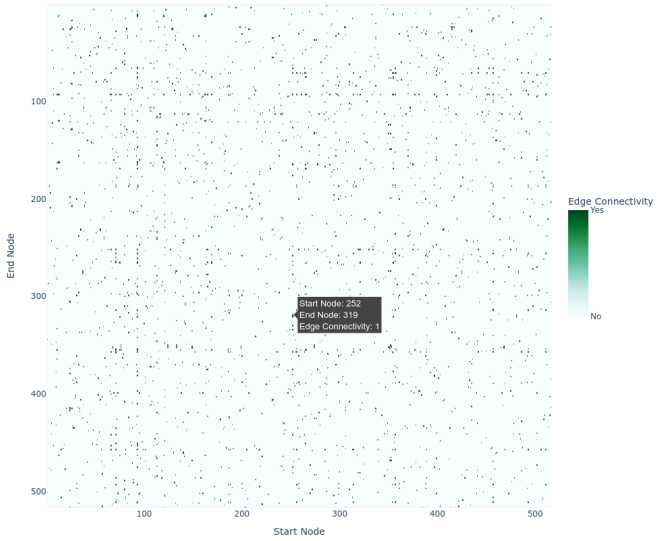

Fig. 2. Full matrix view - Shows the adjacency matrix for the network, with the black colored cells denoting existence of an edge.

C. Matrix Visualisation

For generating the matrix visualisation of the node-link diagram I used the Plotly library of python. The approach was to use a heatmap and use a custom colorscale. Since each value of a cell can either be 0 or 1, I set the minimum value for the colormapping as 0 and the maximum as 1. This way the final visualisation generated shows the dark cells which confirm the existence of an edge between the nodes. The important thing to do was ensure that the cells which have value 1 can be distinctly visualised in the screen space because a 516*516 grid results in very small grid cells. Coloring the cells with an edge with a dark color enables the same. In Fig 2 the entire matrix can be seen. Furthermore when mouse is hovered over the cell the start and end node along with 0/1 value is shown. Fig 3 shows a zoomed in version of the same matrix. A user can zoom into a specific region and view the edge connectivity in that region.

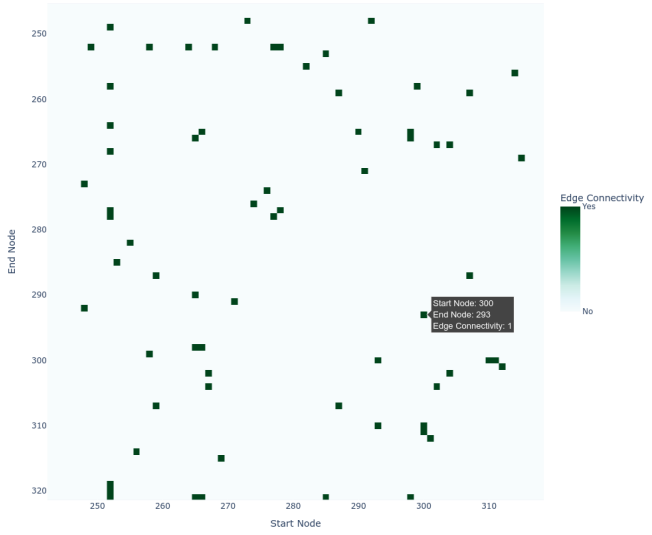

Fig. 3. Zoomed matrix view - Shows a part of the adjacency matrix for the network, with the black colored cells denoting existence of an edge.

IV. Multivariate Data Visualisations

There are two different visualisations that I have generated for the multivariate data visualisation - a treemap and a parallel coordinates plot. It was important to realise the type of data provided. The categorical data is not completely independent and hence must be approached at carefully while creating the visualisations.

A. Parallel Coordinates Plot

For making the parallel coordinates plot I used Plotly. Furthermore since the data that we have includes categorical division it was better to draw a parallel coordinate plot with brushing. This is achieved in Plotly by using the `Parcats` class of Plotly meant for categorical visualisation. As can

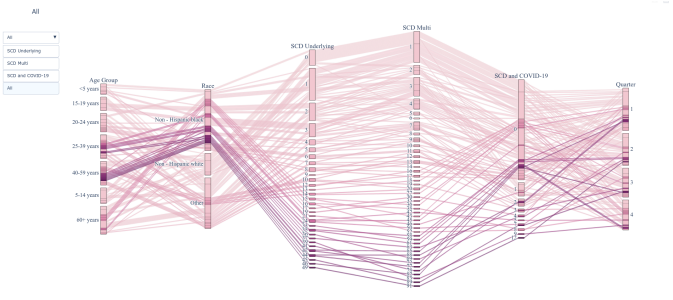

Fig. 4. A parallel coordinates plot with brushing showing all the numerical values along with three categorical values

be seen in Fig 4, there are several options for selecting the numerical column of the data(SCD and Covid-19 deaths, SCD as Underlying cause of death, and SCD as Underlying or Contributing causes of death) that we want to be shown on the plot. This visualisation can be helpful in concluding the relation between the different categories. Since the axis can be moved around, it makes it easier to make conclusions between different data-columns.

B. Treemap Visualisation

For the treemap visualisation I have generated two types of visualisations. I have used both D3 and Plotly. In Fig 5 a user can select the category upon which to perform the level 1 splitting- Age Group, Race or Hispanic Origin or Quarter. The other option determines the numerical value column to be used for splitting the boxes in each of the parent category.

Though this visualisation provides a good enough information about the number of deaths in different type of categories, it is not that useful. The internal boxes though have been divided on the number of deaths but without another level of hierarchy this information alone is useless. Another fault with this comes when we look at the Quarter information. Each year is divided into 4 quarters and we have data from 2019 first quarter to 2021 first quarter. We know that effect of Covid-19 was different across all the years depending on the lockdowns, introduction of vaccines etc.

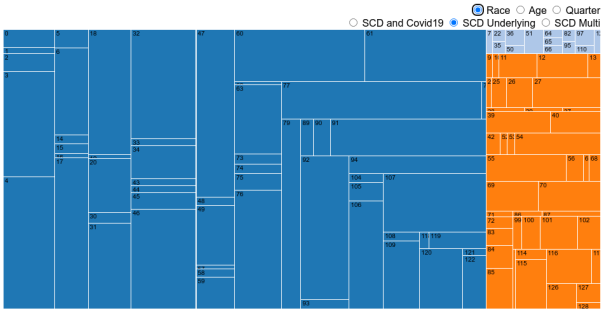

Fig. 5. Treemap visualisation with option to change parents and the values for the categorisation

Hence I created a cascaded treemap visualisation(Fig 6) which would first divide the data based on the years and then splits on the basis of the quarter. This way one can see clear differences between the different years firstly with the increase in deaths in 2020 as compared to 2019. Futhermore, the division of datapoints based on the quarter now makes more sense and provides better information. The data has further been further divided on the races first followed by the gender group.

V. Results

I was able to generate the different visualisations for both a network bsaed datasets and the multivariate dataset. I was able to manipulate the algorithm used for creating the force directed graph and make changes to the visualisation based on the type

Fig. 6. Cascade treemap visualisation generated based on the year, then the quarter followed by race and age.

of data provided. Furthermore, I was able to conclude the importance of the different visualisations for multivariate data - parallel coordinates plot and treemaps and the advantages and disadvantages of different treemaps.

References

[1] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: https://networkrepository.com

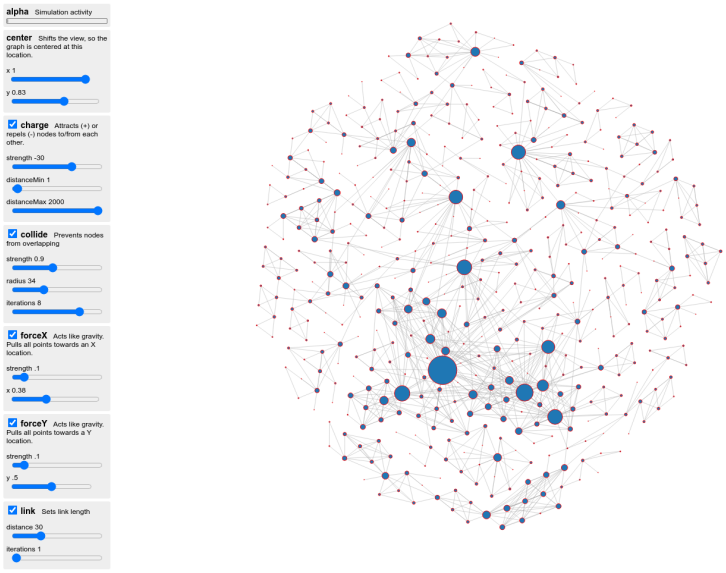

Fig. 7. A Force Directed Graph with node size dependent on number of connections and centering gravity force enabled

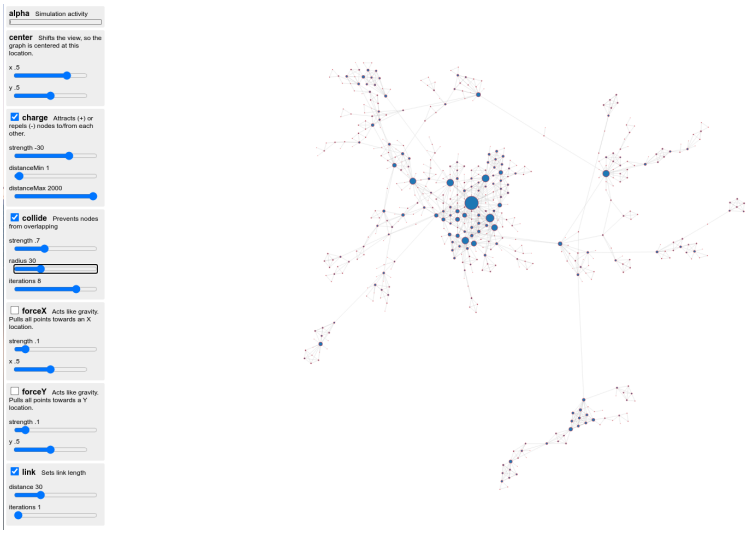

Fig. 8. A Force Directed Graph with node size dependent on number of connections and no gravity