1. Loading Datasets

For my development, I was more concerned about the layout and drawing graphs. Importing new datasets was lower on the list. Currently datasets are stored in JS files as var network = {...}; These JS files are then linked directly into the html and called into the display through a dropdown selection. The function that does this is called ReadData().

1. Drawing New Graphs

The DrawGraph() function draws graphs given it's arguments, nodes and links, which are each an array of JSON objects. The function starts by setting the currentGraph variable to the new nodes and links, giving the node objects an initial position near the center of the screen, and by clearing the previous graph. The links are the first to be drawn. Directed graphs are drawn by using <path> elements, whereas non directed graphs are drawn using <line> elements. If the graph is weighted, this is where the lines are set their thickness. Next, the <circle> elements for the nodes are drawn. Finally, a force simulation is created that binds the <circle> elements to the <line> or <path> elements. The simulation is the restarted and the graph is drawn.

1. Updating Graphs

The force simulation takes care of updating values for x and y in the node and link objects. All that needs to be done in the update function is to take those values and assign the html elements to them. The <circles> have cx and cy which are updated, and the <lines> have x1,x2,y1,y2 that are updated. Directed graphs are different however because they are drawn using arrows instead of simple lines. For each link object, the function iteratively identifies if the link has a double. In other words two nodes both link to each other in two different links. This is important to know because the update needs to know whether to draw a <path> with arrows on one side or on both. If the link has a double, the variable doublePath is set to true. Next, the update takes the two points provided by the force update (x1,x2,y1,y2) and turns that into six points which are used to draw the arrows. The diagram below explains the calculations of these points. With these points set as variables for reference, the update creates a string that is used as an argument to draw the path function. All or only some of the points are used depending on the value of doublePath in order to draw the correct kind of link between the two nodes. Similarly, if arrows are turned off in the UI, a simple line is drawn between P1 and P2. After the string argument is returned and the path is drawn, doubled paths are tagged and destroyed in order to prevent duplicate html elements and forces in the force simulation.

P1

P2

P3

P4

P5

P6

a

b

P1’

P2’

θ

ɸ

ɸ

L

L

1. Select Node

SelectNode() takes an argument nodeId, creates and draws it's egonet, and initializes the graph. Different elements in the html UI call upon this function either by giving user input as the argument, or by providing a random nodeId. After finding and drawing the egonet of the nodeId, the function recolors the ego node and fixes it in place. Also, depending on if graph updates are turned on, the other nodes in the graph are initialized as fixed or free. Finally, the force simulation is restarted.