**Network Visualization Front-End**

To build the front end of the network analysis website, the bootstrap framework was used. The basic version of Bootstrap was used (i.e. not with Source or Less). The advantage of using Bootstrap was that no media queries were necessary and all the graphs and navigation bar scaled easily. Any future graphs or analyses that needed to be added to the website could be also done so easily without needing to modify the entirety of the code. Simply adding another container, column, row and graph is all that is needed.

*<div class=”container”>*

*<div class=”row”>*

*<div class=”col-lg-12 col-xs-12”>*

*<div id="analysis#"></div>*

*</div>*

*</div>*

*</div>*

For the navigation bar, simply adding another link and changing the CSS of the body to allow more space between the body and navigation bar is all that is needed.

*<li class="link" id="analysis#link"><a href="analysis#/analysis#.html">Analysis #</a></li>*

*.body {*

*margin-top: …;*

*}*

To create the graphs, some back-end code was needed to properly extract the needed data from the graph and generate the data as json objects. Once that was done, Plotly was used to generate the graph to put in the container. To use Plotly, the script had to be called.

*<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>*

In a separate javascript file, four functions were created:

*loadJSON(file, callback)*

*returnLayout(title, x\_axis\_title, x\_axis\_type, y\_axis\_title, y\_axis\_type, hovermode)*

*returnLayout2(title)*

*load(jsonFile,divID,layout)*

The returnLayout and returnLayout2 functions take the string values to create the axes and the location of the legend for each graph. returnLayout is for x-y graphs and returnLayout2 is for non-x-y graphs like a pie chart. An object is created and the object retains these values. This object is passed onto the load function. The load function calls the json file:

*loadJSON(file, callback)*

and parses through the values. Each value is passed into an object. Since the values are already cleanly generated through the back-end code, it is not necessary to do the parsing again. The Plotly function:

*Plotly.newPlot(divID, actual\_JSON,layout)*

is used to generate the graph.

The div which the Plotly will place the graph in will need to have an id for which the Plotly function takes in as one of its parameter. It should be noted that the Plotly function will generate two nested divs within the div specified.

*<div id=“analysis#”>*

*<div>*

*<div>*

*</div>*

*</div>*

*</div>*

All that is necessary is to write in the html file:

*<script src="../scripts/plotFunctions.js"></script>*

*<script>*

*var layout = returnLayout(…);*

*load("../json/****necessaryFile.json****","analysis#",layout);*

*</script>*

necessaryFile.json is the json file containing the necessary data. The above code is written after all the scripts.