# Counterparty Visualization categorized by Rating

The visualization is hosted [here](https://deepthiyathiender.github.io/Dendograms/Dendogram.html). The page consists of two visualizations – a Dendogram and a Pie Chart. Both visualizations are built using D3.JS.

## Dendogram

In order to build the dendogram, the program requires a JSON tree of the nodes to display. The JSON tree is built using [this script](https://github.com/deepthiyathiender/Dendograms/blob/master/ToTree.html). The script to generate the tree is not a generic one and will need to be tweaked each time a new tree has to be built and will depend on the depth of the tree. See below for a snapshot of the tree that is generated.

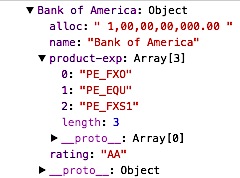


Each node has to mandatorily contain the following JSON object properties – a name, a parent and children (if it has any children). If these property names are buggy, the dendogram will not render. For a more detailed explanation on how to build dendograms in D3, check [this](https://bl.ocks.org/mbostock/4063570).

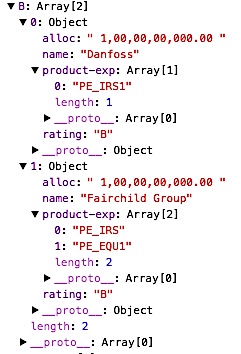
The root of the tree (in this case, the root is Rating) will set its parent to null. Each of the children is JSON objects and follows the same properties pattern – name, parent and children. In addition to these mandatory properties, we can also bind additional data that each node should require (in this case, we bind the alloc property for each counterparty; this alloc property is later using when calculating the utilization).

The JSON tree is nothing but a JavaScript array of Objects. The only difference is that the array is serialized meaning all strings are quoted. Check [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify) on how to serialize JavaScript objects to JSON.

[The script](https://github.com/deepthiyathiender/Dendograms/blob/master/ToTree.html) to convert to JSON performs three primary functions – loads the CSV into the browser using [d3.csv](http://learnjsdata.com/read_data.html), builds one or more hash maps and generates an array of JavaScript objects. The primary CSV used to build the dendogram is [here](https://github.com/deepthiyathiender/Dendograms/blob/master/creditLimit.csv). Using this CSV, we would like to build a hashmap hashed over Counterparties. A counterparty name points to – its rating, its allocation, an array of its product exposure. See below for a snapshot of a hashed counterparty.



In addition, we also create a hash map hashed by rating. See below a snapshot for that.



The Rating pivoted hash map contains an array of counterparties based on their rating.

Once the two hash maps are built, we use them to build the tree of objects. In order to build trees, this function is what needs to be tweaked. It is basically several nested for loops – the number of for loops is the same as the depth of the tree we need. Each iteration of the for loop pushes an object (name, parent, children) on the array. Its best to work bottom up; build the tree from the leaf nodes up to the root. At the end of the nested for loops, the JavaScript array needs to be “stringified” or “JSON-ified” using the JSON-JavaScript Stringify method.



The JSON tree can then be copy-pasted [here](https://github.com/deepthiyathiender/Dendograms/blob/gh-pages/Dendogram.html) into the “treedata” variable.

If the tree has been built correctly, you should see the dendogram generated.

## Pie Charts

[D3 Donut Charts](http://codepen.io/adeveloperdiary/pen/jbdMKr) are used to display counterparty’s funds utilization. Each time counterparty in the dendogram is clicked, a donut chart of that counterparty’s utilization is expected to be seen.

Each counterparty invests in one or more of these instruments - [Bonds](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_Bond.csv), [Equities](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_Equities.csv), [L&D Deals](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_L%26DDeal.csv), [FxOTC Deals](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_FxOTC.csv), [FxSwap Deals](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_FxSwap.csv) and/or [IRS Deals](https://github.com/deepthiyathiender/Dendograms/blob/master/mega_table_IRSDeal.csv). Each of these instruments (its data) is maintained in different CSVs (check hyperlinks above). These tables cannot be merged into a single table because the tables have different columns and its better to separate the concerns. We create a separate hash map pivoted around counterparty for each of the instruments.

When a counterparty in the dendogram is clicked, depending on its product exposure array, we hash into one of the instrument hash maps to retrieve the specify values for the amount of funds utilized for that particular counterparty and that particular instrument. Using this information, we calculate the utilization ratio and render/update the donut chart.

The key takeaway is creating the different hash maps and knowing which one to access when a node in the dendogram is clicked.