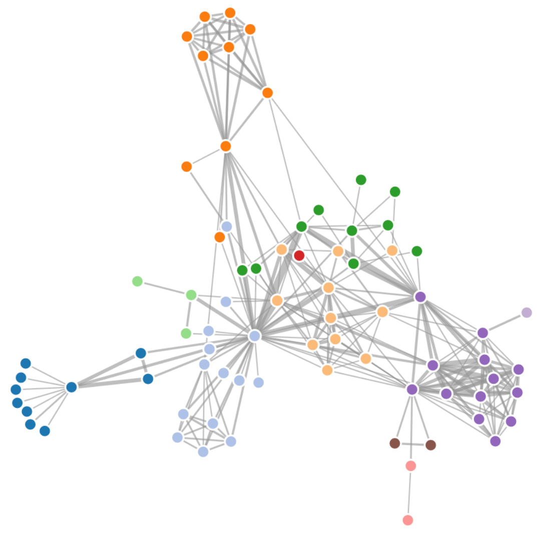
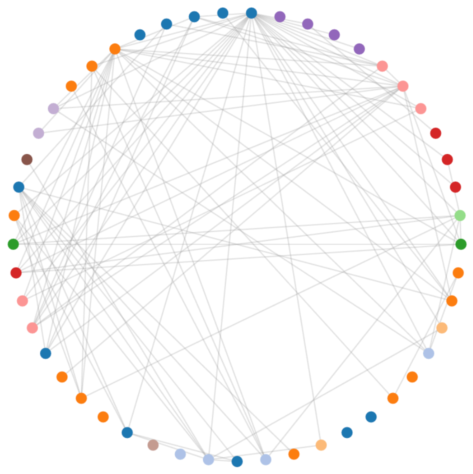
A quick exploration of the data reveals that it is composed by nodes and links (edges), hence it is intuitive to think of a force-directed graph as a first option. As a first draft I coded a straightforward implementation of this type of visualization using the Javascript library d3.js, which required me to generate two CSV files: one for the nodes and one for the edges.

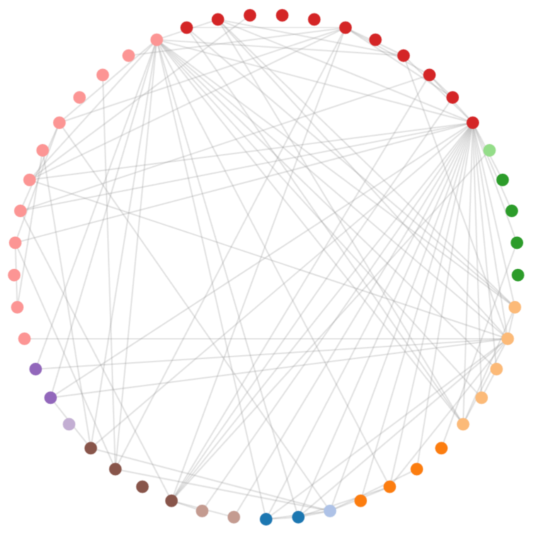


Here each node is color-coded by the attribute “Object” and links are drawn based on the information of the columns “source” and “target” in the edges.csv file. Although the result was nice, it felt a bit messy, so it needed some refinement.

I started thinking about a better way to distribute the nodes spatially. At the same time, I wanted to make the graph more aesthetically pleasing. The solution for this was to locate the nodes in a circular layout, which resulted in a visualization like this.

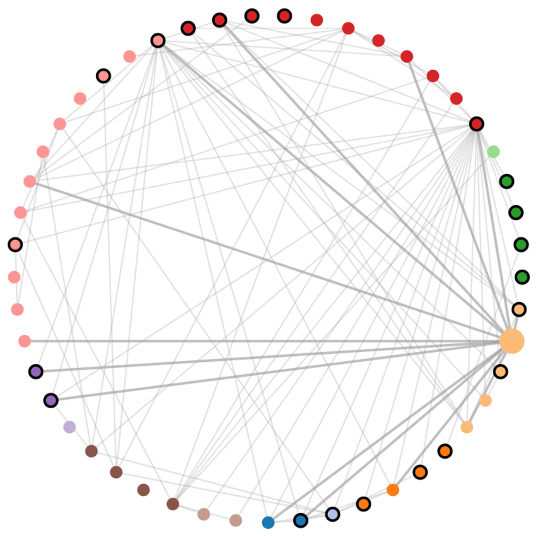


This is already looking much nicer and more appealing, but the colors (based on the attribute “Object”) do not seem to follow a particular order, which makes the user harder to see different groups in the data. To make it easier for the user to identify groups of nodes of the same “Object”, I decided to sort the nodes based on that attribute before plotting them.



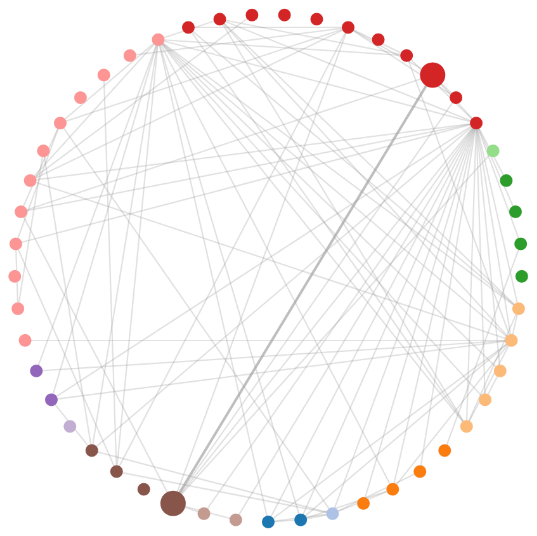
This new version guides the viewer to identify groups of “objects”, and hence it requires much less effort from the user.

With the basic layout already set, it was time to start adding more details and interactivity. First of all, there are two many lines, which might be confusing for the user. In that case, it would be nice to have a tool to clearly see how one once is connected to the others. For that I used a “mouseover” function so that each time the mouse is over one of the nodes, it highlights that node and also the links to/from it.



The mouseover technique is flexible enough that allowed me to modify other attribute of the nodes at the same time. For instance, I highlighted nodes that shared the same “Activity” by changing the stroke of the circles. In summary, we can see the connections and all the other nodes that share the same “Activity” just by hovering over a node.

A similar implementation can be used for edges. In this case, every time the mouse is over a link I highlighted the nodes connected by such link.



There is still a lot of information that can be added to our visualization. For example, we could show the Type/Label of one edge when the mouse is over. Alternatively, we could change the stroke of the link based on the “Label” attribute (solid, dashed, dotted, for example). Another variations might include sorting and coloring the nodes based on “Activity" instead of “Object”, but that would depend, in principle, on what is the main point we are trying to communicate.