TOOL USED: D3 VISUALIZATION
Why D3?

1. **D3 is for visualisation**, as opposed to plotting. The basic concept is that you create a collection of objects (usually shapes in an svg), and bind data to them. You use the data to manipulate their properties, so you can have e.g. rectangles positioned around the screen based on a vector of input data. D3 is a bit painful for making simple plots, but it really shines when you want to do something a little more custom.
2. **D3 is great for animation and interactivity:** Animation is almost trivial with D3. Remember those rectangles? Simply bind a new vector of data to them, and use the built in transitions to make the shapes slide pleasantly into their new positions (or sizes, colours, etc.). Interactivity comes from the fact that D3 is implemented in javascript, and allows you to make these transitions occur in response to keyboard or mouse input.
3. **D3 is magic**, in that I can send anyone in the world an html file, and the visualisation will render perfectly in their browser. The only caveat is that it has to be a browser that complies with HTML5, CSS3 and embedding javascript (which makes any recent version of Firefox, Chrome, Safari or even Internet Explorer 9+ fine).

INPUT: clusters.rsf file generated from RELAX output

OUTPUT: Visualisation diagrams showing the hierarchical structure and also the components containing entities.

What is a hierarchy layout?

A hierarchy layout is an abstract layout which is generated from data having a clearly defined parent-child relationship/s. The base hierarchy layout in D3 allows the library to share common code between different graphic layouts, all descendants and observing a hierarchical layout. These descendant layouts are discussed in this topic.

But what do I mean by "common code"?

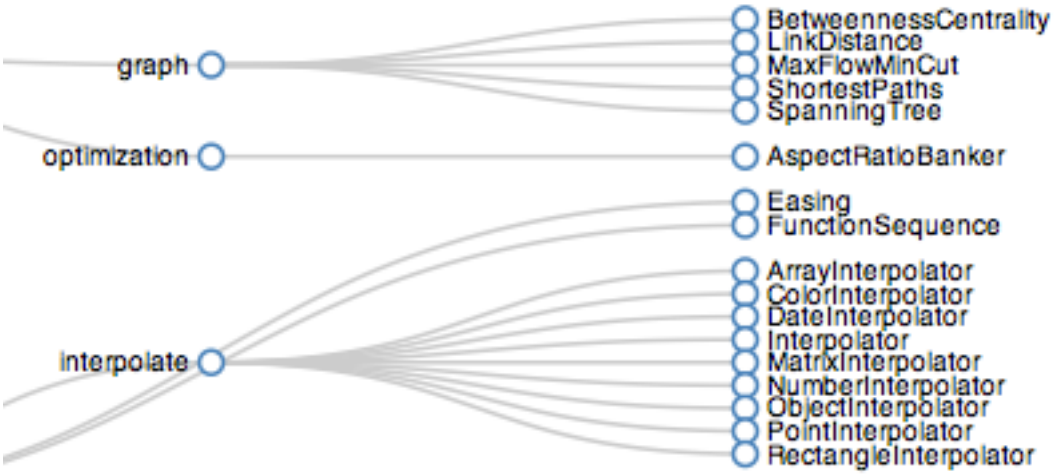
A hierarchy layout is bound to a data object. Certain properties are common to all hierarchy layouts which is why the base layout provides configuration options. For example, the base hierarchy layout answers the following questions:

1. Which property in the bound data object represents the children?
2. Which property in the bound data object represents the value we wish to display?
3. How should we order the data?

The available hierarchical layouts are:

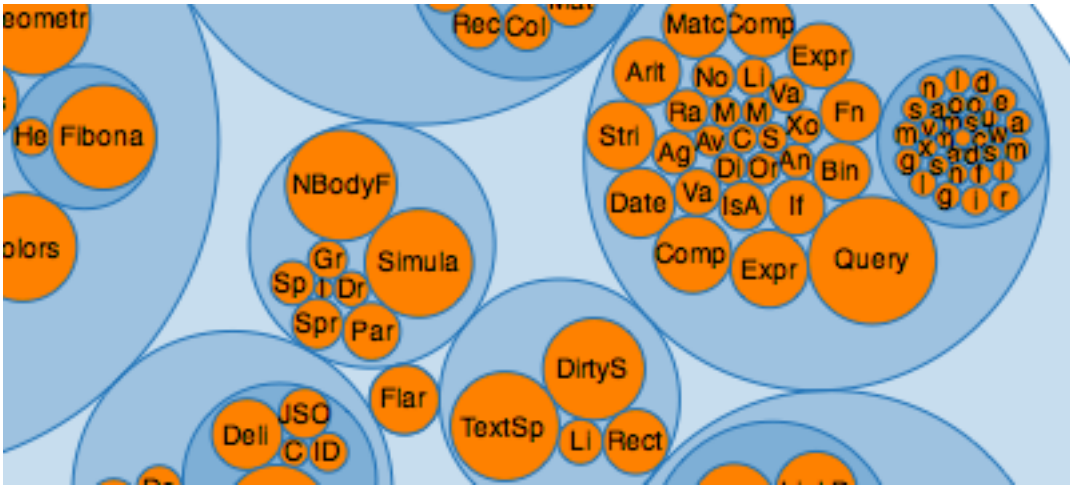
– *Cluster Layout*
– *Pack Layout*
– *Partition Layout*

Cluster Layout



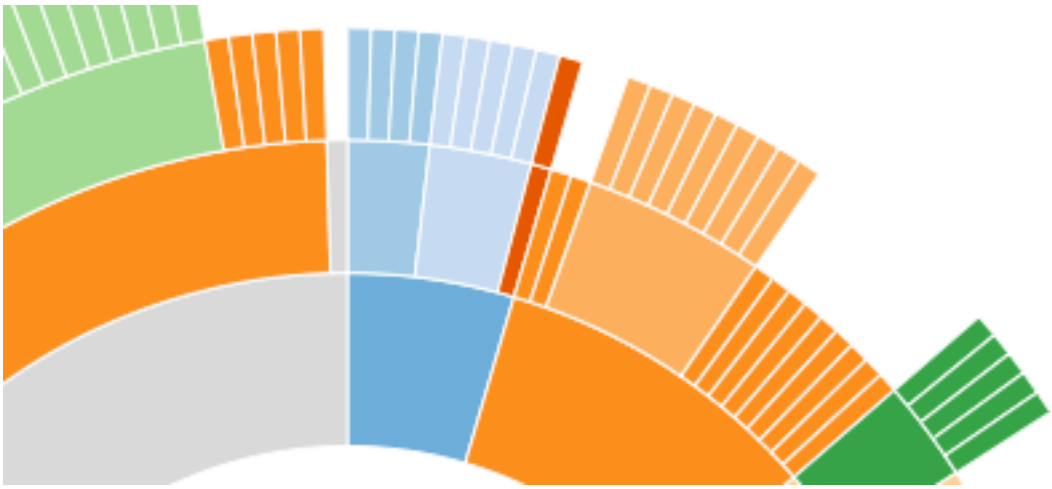
A cluster layout is a diagram representing some kind of cluster. Dendrograms are produced by this layout which places all the leaf nodes at the same depth, i.e. all the final children (the leaves of the data tree parent-child relationships) at the same level. Clustering at higher levels in the hierarchy are displayed at varying distances according to dissimilarity.

Pack Layout



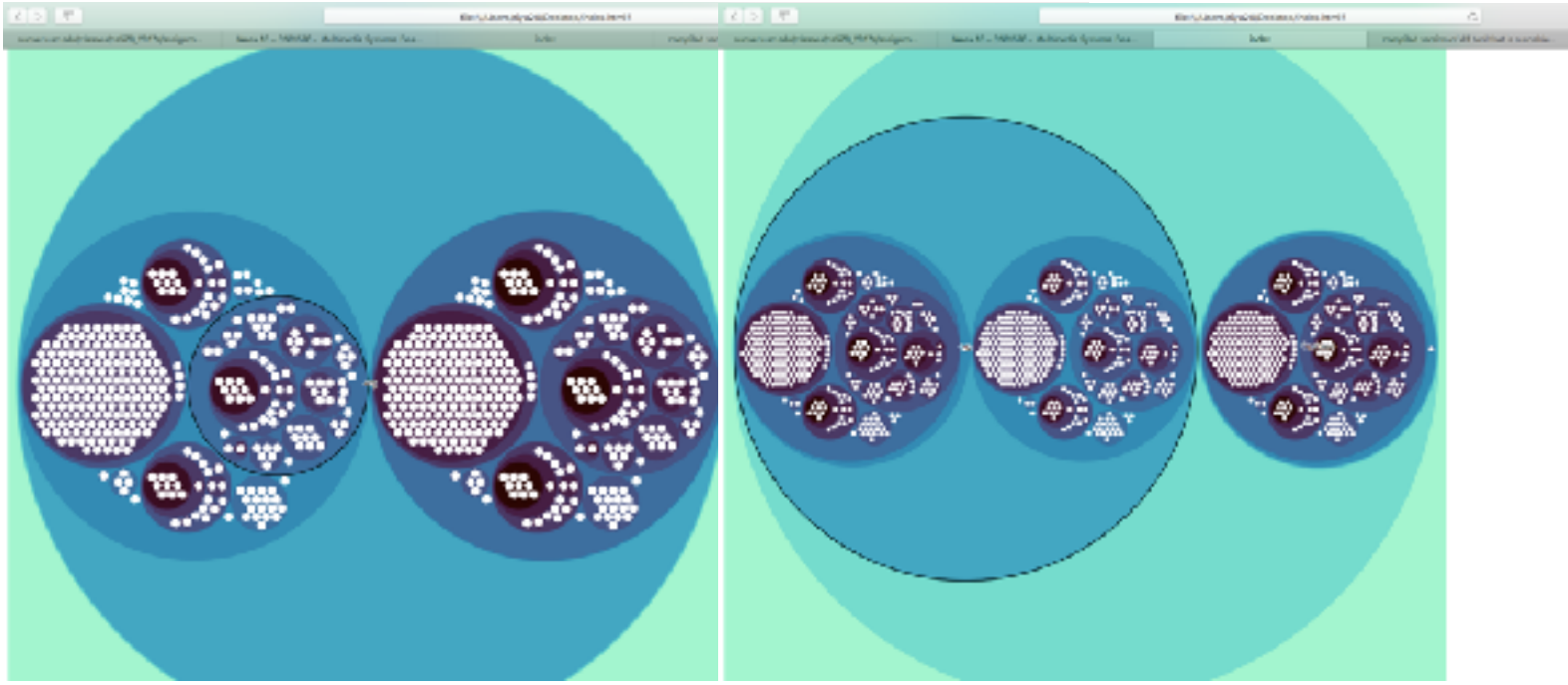
A pack layout is an enclosure diagram which uses containment to represent the hierarchy. An added feature of this layout is that leaf node size can be linked to a quantitative dimension of the data point. Each grouping is enclosed in a circle however due to wasted space only the leaf node size can be accurately compared. This type of layout doesn't make the best use of available space and therefore undermines its efficacy. A pack layout is still an engaging and entertaining layout and is still used for many applications.

Partition Layout



The partition layout produces an adjacency diagram. The size of each node can be used to represent a quantitative dimension of the data which would not be visible with any of the node-link layouts. Nodes are drawn as solid colours and placed adjacent to their parent node, there are no links drawn between parent and child.

Example:



apache 2.3 apache 2.4

The above circle packing visualisation shown is generated based on the input provided (cluster file) which works on some values on input data.
Though working on a part of input, it shows the

1. what the component (“graphics” in the example) consist of?
2. how the two systems contain files in a different structure.
3. Also, files which may be same have a complete different location and the files they include also differ.

4. Also, the size and the composition of the entities can be understood, (.org in the above example)
5. As the diagrams show the hierarchical structure of the system, it relates to the the composition and the clusters of entities formed depending on the components.

These diagrams can scale to different system sizes, if input used is of the type of “more than 300”, as specified in the readme file.
Since this takes the entire file as input and groups them in various packets or categories showing the composition of components in the system.

Overview of the system:

In short, the system is represented in terms of hierarchy of the components, along with their composition and representation, also cluster fragmentation seen and drilling down to the specific cluster of interest possible
You can view the further categories in a particular entity, by zooming in or out, clicking a specific category (interactive diagrams), the interactive element is to aid the exploration of information
Also, the differences can be made based on the zoomed out version of the diagram which depicts various colours and sizes depending on the category.
The dependencies and subsampling of the diagrams show the connection between the components.