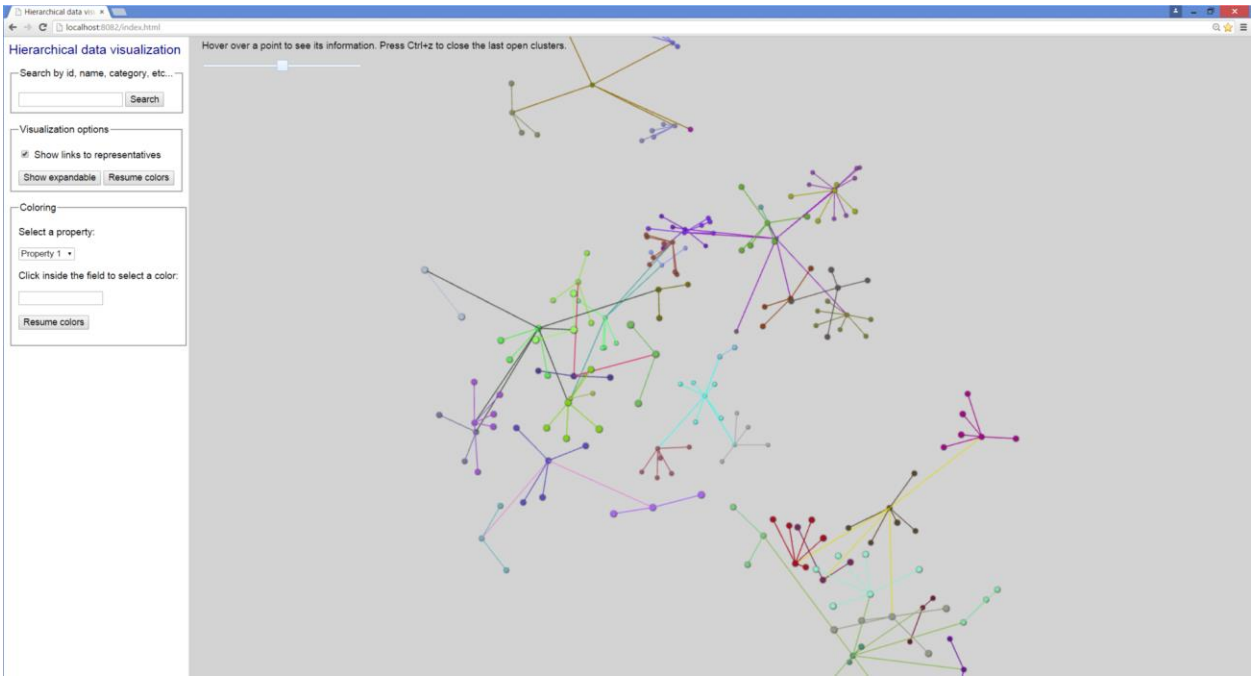
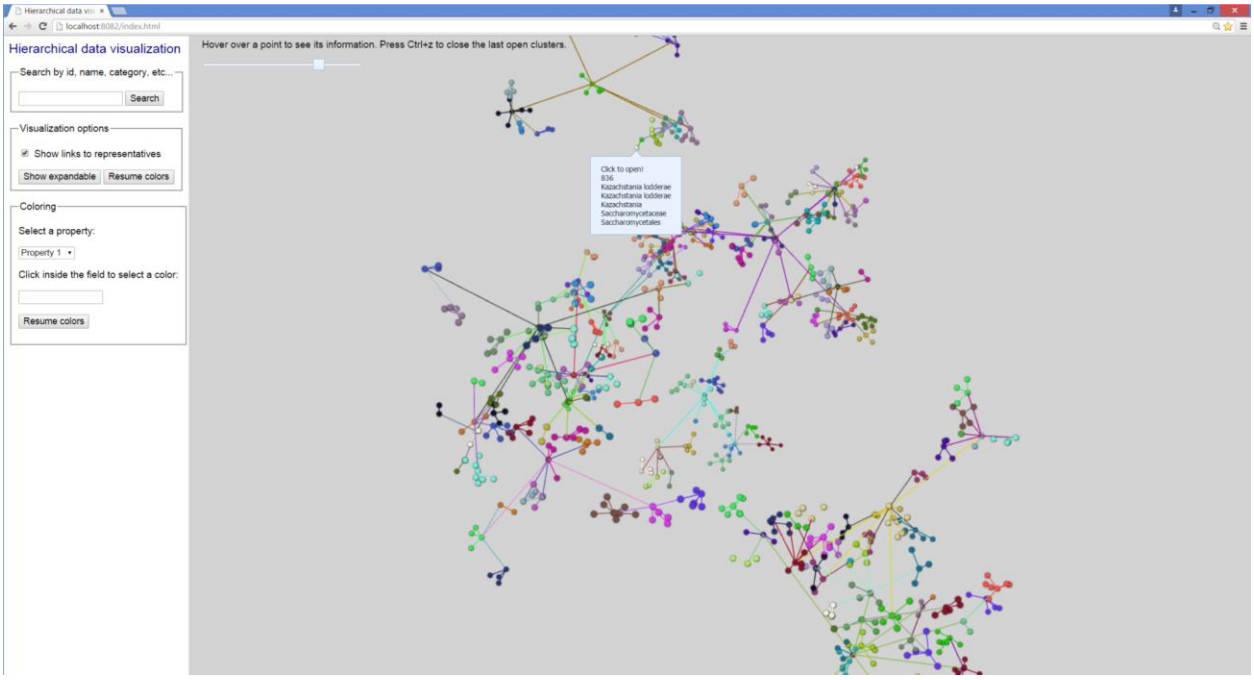
H3dvis – Hierarchical 3D data visualization: User & developer manual





The purpose of the web application H3dvis is to enable 3D visualization of data in a hierarchical manner. In many scientific applications data is hierarchically clustered. The scientist would be then interested to view the clusters by expanding and collapsing the clusters.

A cluster is expanded by clicking on its “representative” point. This is a pre-selected point that is central to the cluster.

The application also enables to colorize points based on features other than their clusters.

Data description and functionality

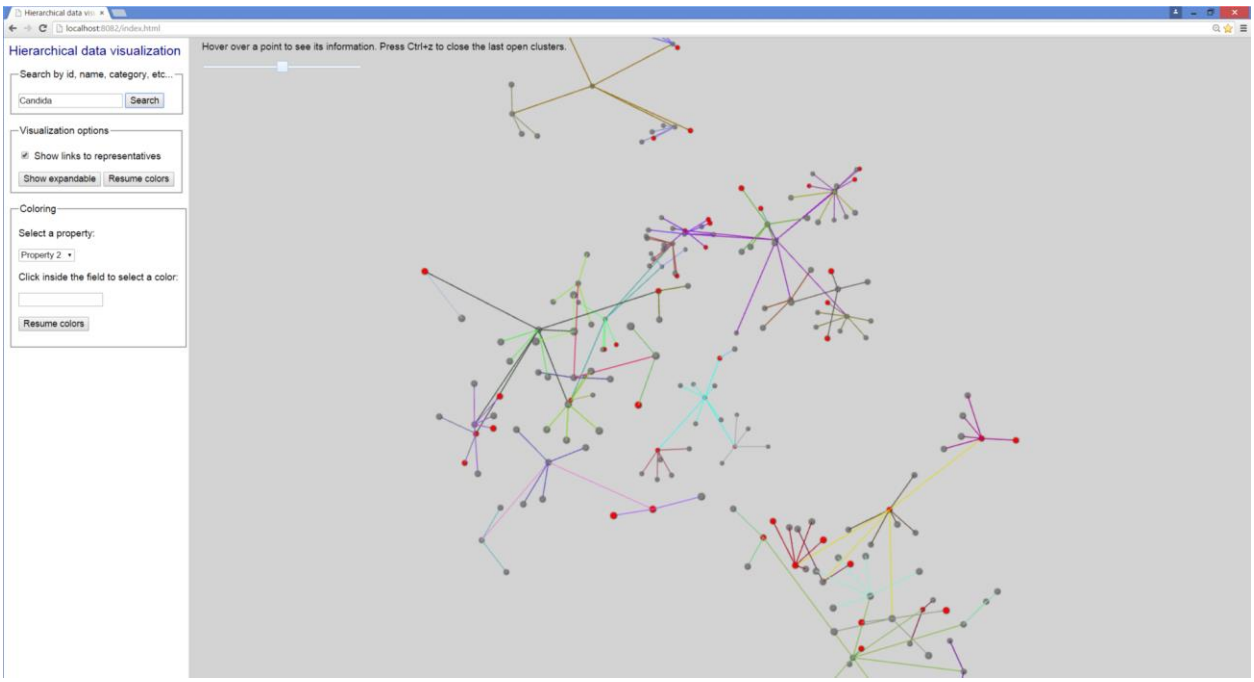
Every point has 3 coordinates and a unique ID. It also has an associated *path,* that contains its preceding representatives in the clustering hierarchy. For example, if point with id 87 has path 398.33.56.21, this means that at level 0 of the clustering hierarchy its representative is the point with id 389, then at level 1 its representative is 33, at level 2 its representative is 56, and at the last, level 4, its representative is 21.

A point can optionally have “Categories” and “Properties”.

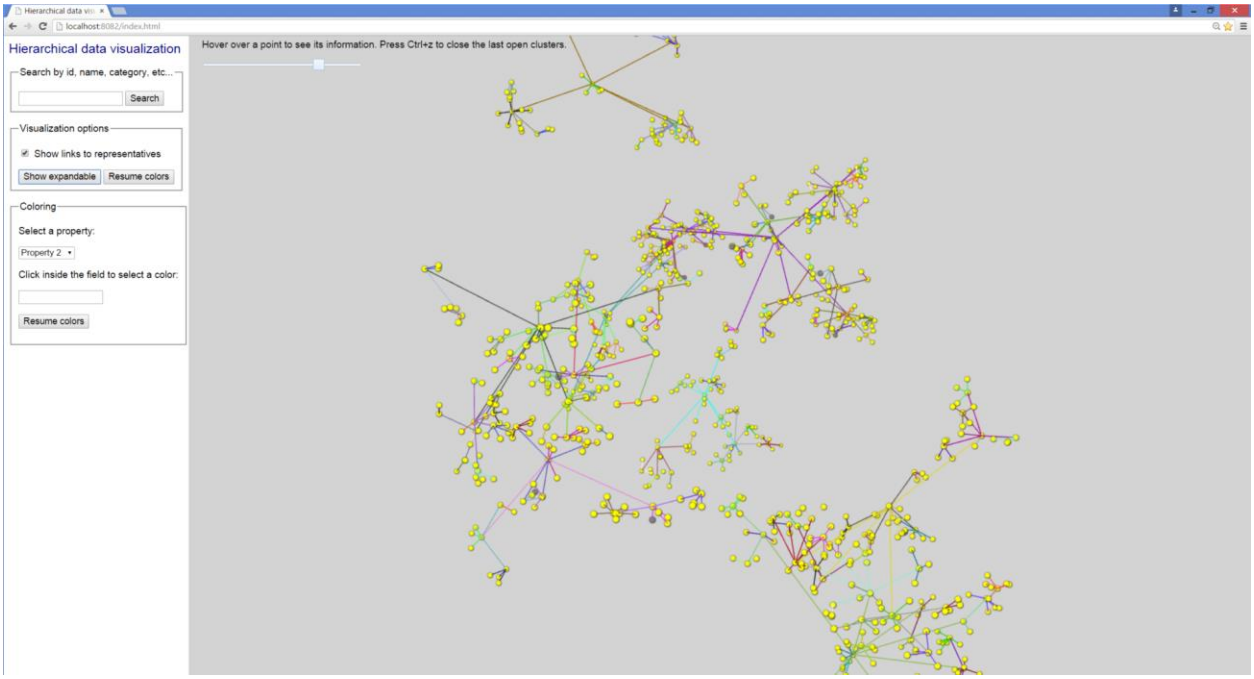
“Categories” is a list of names associated with the point. This list is displayed when a user hovers over a point with the mouse or equivalent.

“Properties” is a list of real numbers. Each number represents the intensity of a respective property. These numbers are used in the *Coloring* section of the UI of the web-page. When the user selects a property, and a color, every point is colored with a shade of the selected color. The intensity of the color corresponds to the intensity of the selected property for the particular point.

A user can search for all points that contain a certain substring in their ids, names or categories, by using the *Search* section. Then all points that are a match become red, and the rest become grey.



By clicking the “Show expandable” button in the *Visualization options* section, all points that represent non-singleton sub-clusters in the subsequent levels are colored yellow. The rest of the points are colored grey.



The “Resume colors” buttons return the colors of the points to the previous coloring scheme.

The software can run in two modes, *big data* or *small data* mode. These will be explained in the next section. When the mode is *small data*¸ a slider appears on the top of the screen, that enables the user to load a whole new level of clustering at once, rather than clicking on the points.

Data format

All data is in the sub-folder *data* of the main folder. The hierarchy of clusters is stored in a corresponding hierarchy of folders. Each folder contains a file data.json, which contains all information for the points of a particular cluster. If a point in this file is representative of other points in the following levels, then there must exist a folder that has the same name as the point ID. For example, assume that we are in folder data/2/21. Assume that in this folder there are three folders with names 21, 62, and 89. Then, in the file data.json that is in folder data/2/21, there must be information about the points 21, 62, and 89. In addition, it must hold that the path of point 21 begins with 2.21.

Directly in the folder *data*, there must be at least two more files , NamesOfProperties.json, and MetaData.js.

NamesOfProperties.json includes a list of the names of the properties of the data, in a JSON format.

MetaData.js defines whether the software is running in a *small data* or *big data* mode. It must contain one of the lines

var bigData = false;

or

var bigData = true;

If bigData == false, then this means that the software is running in a small data mode. In this case, there must be a file data/smalldata.json. The data in this file has the same format as the data in every data.json file (described below), but the file smalldata.json contains *all* data.

It is not strictly defined how much data means big data, because it depends on the data itself, and its distribution. The example (small) data contains 4271 points.

Next, we explain the format of the contents of the data.json files in every folder in the hierarchy. The data is in a **JSON** (JavaScript Object Notation) format.

To obtain data.js, first a data structure Dictionary<string, Point> is created in any programming language (see for example https://msdn.microsoft.com/en-us/library/ekcfxy3x(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1 for the dictionary data structure), where the keys are the id's of the points and Point is an object of the class

```
public class Point
    {
        public List<string> Path;
        public List<double> Coordinates;
        public List<object> Categories;
        public List<double> Properties;
    }
```

Path, Coordinates, Categories and Properties are as discussed in the previous section.

Next, the dictionary is serialized using JavaScriptSerializer and written in data.json.

Here is an example of an entry of the serialized dictionary in a data.json file:


```
"3951":{"Path":["3951","3951","3951","3951"],"Coordinates":[0.998608003838931
67,0.61276015046241838,0.450976426942296],"Categories":["Prototheca
cutis","Prototheca cutis","Prototheca","",""],"Properties":[9,4,4]}
```