**CSCE 5320 Scientific Data Visualization**

**ICE-8**

**Visualization of Spatial Data, Networks, and Trees**

# 1. Making Maps(45 points)

Using TOPOJSON data links provided in the tutorial and making maps in d3.

Firstly, I have loaded the TOPOJSON for the world map to the git hub gist, with the file name ‘world-atlas.json’. This JSON file has topology information about all the continents and countries in the world.

Raw data is as shown below:

Now I have written the code to display the world map on VizHub.

### index.html:

Firstly, we create an HTML file that will be used to display the world map. We use a link tag to link the styles.css file, which has CSS defined for the webpage. We also link the JavaScript files to make use of all the D3.js and TOPOJSON tools. We use the body tag to display the SVG image of the world map in the body, using the index.js file.

### index.js:

This file has the main code for creating the world map. Firstly, all the required functions and packages from topojson and d3 libraries are imported. Then we select the SVG element from the HTML code using the select function. We are also defining the height and width of the SVG image. We then create a header text to display the header, by appending it to the SVG element. The ‘geoNaturalEarth1’ function from D3.js is then called to create a projection for the world map, and the ‘geoPath’ function is used to create a path generator for the map. We are using the ‘scaleOrdinal’ function and the color scheme ‘schemePastel1’ to display each country in a different color, we are using a pastel color pallet here. We Append the path element to the SVG element and determine the shape of the map as a sphere.

Finally, we load the JSON data from GitHub and convert it into a feature collection for different countries. For each country, path elements are created, class is set, and path data and color fill for each path are set based on countries.

### styles.css

This file defines the style of the Webpage, by defining the background color and other parameters of the webpage.

Below is the image of the world map.

**GitHub gist:** <https://gist.githubusercontent.com/nehabaddam/6cd52d69434b442213e9cc20c8610e9c/raw/1f2cf89618f7db82a2442abb7a94e11f70f418df/world-atlas.json>

**VizHub Link:** <https://vizhub.com/nehabaddam/ddbdba4659cc43aea66e32f3187b48c3>

**VizHub ID:** nehabaddam

Firstly, I have loaded the TOPOJSON for the Australia country map to the git hub gist, with the file name country.json.

### index.html:

Firstly, we create an HTML file that will be used to display the world map. We use a link tag to link the styles.css file, which has defined CSS for the webpage. Now we link JavaScript files to make use of all the D3.js and TopoJSON tools. We use the body tag to display the SVG image of the world map in the body, using the index.js file.

### index.js:

This file has the main code for creating the Australia map. Firstly, all the required functions and packages from topojson and d3 libraries are imported i.e., select, JSON, CSV, geoPath, geoAlbersUsa, geoMercator, scaleOrdinal, and schemePastel1 from D3.js and feature, mesh from TopoJSON. By using the select function we get the SVG element from the HTML code. We are also defining the height and width of the SVG image. We then create a header text to display the header, by appending it to the SVG element, the header text displays ‘Australia Map’. The ‘geoMercator’ function from D3.js is then called to create a projection for the Australia map, and the ‘geoPath’ function is used to create a path generator for the map. We are using the ‘scaleOrdinal’ function and color scheme ‘schemePastel1’ to display different states in different colors, we are using a pastel color pallet. We then Append the path element to the SVG element and determine the shape of the map as a sphere.

Finally, we load the JSON data from GitHub and convert it into a feature collection for different countries. For each state, path elements are created, class is set, and path data and color fill for each path are set based on countries. On the whole, we use this code to generate a map of the country Australia with different state divisions colored in different shades.

### styles.css

This file defines the style of the Webpage, by defining the background color and other parameters of the webpage.

Text

Description automatically generated

Below is the image of the country map of Australia.

**GitHub gist:** <https://gist.githubusercontent.com/nehabaddam/0be7470424c9533fcd9541a3dd4b9fde/raw/5ccdd1431b27918a919586dd11df9e61c3e43f6f/country.json>

**VizHub Link:** <https://vizhub.com/nehabaddam/ff8648db2be94bddb198fc126303b0a6>

**VizHub ID:** nehabaddam

### World Map:

I have used the **geoNaturalEarth** function for projecting the world map. The geoNaturalEarth1() function is a built-in function of D3 which is used to create a natural projection for the world map. This projection preserves both distance and direction from the central part of the map and a modified cylindrical projection for the outer regions. The reason for using **geoNaturalEarth** geo-projection is that it provides a good balance between the shape and size distortion of the countries and continents.

### Country Map:

I have used the **geoMercator** function for projecting the country map of Australia. The **geoMercator**() function is a built-in function of D3 which is used to create a projection for the world map from the parameters given. This projection maps the Earth's surface onto a flat 2D plane. The reason for using **geoNaturalEarth** geo-projection is that it provides an easy way of representing the world map that is widely recognized. It preserves the shapes and angles of small features on the map, such as coastlines and country and continent borders, displaying detailed effectively.

# 2. Tree and Network (55 points)

We are creating a Family Tree of Homo Sapiens. Initially I created a tree as shown below.

### data.json

The data file has tree structure data, that defines the parent-child relation between the data. Types are defined to display different colors at each level of the tree. For example, here Hominoideo is the parent of Hominidae.

### index.html

Firstly, we create an HTML file that will be used to display the world map. We use a link tag to link the styles.css file, which has defined CSS for the webpage. Now we link JavaScript files to make use of all the D3.js. We display a header and a container element with an ID of "tree-container" defined. We use the script tags to display the data.json file in the tree format, by creating the boxes.

### index.js

This code is used to create the boxes that represent each node. We code to display each node using a different color based on the types given in the JSON data file. If there is enough data to display the tree, we go ahead and generate a tree structure.

We first fix the root node and then create dimensions for the SVG image by calculating the depth and width of the tree. To the SVG, nodes are added, links are defined, and tooltips are defined. The functions make sure that all the nodes are represented clearly without any collision and the links also represent the parent-child relations correctly.

This code on the whole creates Boxes, that represent the nodes of the tree and colors the nodes based on the type of the node, and adds names to each node.

### styles.css

This file defines the style of the Webpage, by defining the background color and other parameters for the box containing links, and arrows.

**VizHub Link:** <https://vizhub.com/nehabaddam/0e08263c174d45c0b64cd1b0380dd482>

**VizHub ID:** nehabaddam

We have added type to the elements in the tree. Each level of the tree represents a generation. Each generation is represented using a different color. The root node is Hominoideo. The leaf nodes are the bottom nodes, for example, the Homo and Pan.

We are creating a Social Networking Graph with 10 people.

### data.json

This data has Social Network Links between 10 friends, and it shows how they are related to each other.

### index.html

Firstly, we create an HTML file that will be used to display the world map. We use a link tag to link the styles.css file, which has defined CSS for the webpage. Now we link JavaScript files to make use of all the D3.js and TopoJSON tools. We use the body tag to display the SVG image of the social network in the body, using the index.js file.

### index.js

This file has the main code for creating the social network graph. Firstly, all the required functions and packages from topojson and d3 libraries are imported. Then we select the SVG element from the HTML code using the select function. We are also defining the height and width of the SVG image and the color scale for the node. Then we define the node shape as a circle and markers are also defined.

Next, we load the JSON data from the data.json file and define the links and nodes. We are using the ‘forcesimulation’ function to create the links and nodes, events for nodes are defined using the ‘dragstarted’, ‘dragged’ and ‘dragended’ functions.

Finally, we create the links and nodes and set their attributes and events. The behavior of the simulation is set using the tick event.

### styles.css

This file defines the style of the Webpage, by defining the background color and other parameters for the nodes and the links.

Below is the social network that is created:

**VizHub Link:** <https://vizhub.com/nehabaddam/00c793310f724bd4b85c48961375bc7a>

**VizHub ID:** nehabaddam

|  |  |
| --- | --- |
| Network Chart | Tree Chart |
| It shows multiple relationships between elements, without any hierarchy. | It is a hierarchical data structure that offers a single parent-child relationship between elements. |
| It does not have any root node | It has the root node which is the top of the hierarchy. |
| Each element in a network can have multiple connections to other elements. | Each element in a tree can only have one parent. |
| The connections can be of different types, there is no specific direction. | There is a direction of flow from the root of the tree down to the leaves. |
| They are cyclic. | They are non-cyclic. |
| Examples: Social networks or communication networks, internet network | Example: Family tree, organizational trees, species trees |

We use network charts in visualizing the data that shows more than one connection between the elements, where there is no parent-child relation. For example, In a social network of friends, it shows the friends as nodes and connection between them is represented using a link. This relation can be better visualized as a network chart but not as a tree, because people can have multiple friends and those friends may have other friends that are friends of friends. This will form a mesh of friends. A network chart would be appropriate for a social networking chart like this, as we cannot form a tree structure with this kind of data. So, the data that forms a mesh of the network should be visualized using a network chart but not a tree.