12b: Force diagrams

Also, hierarchical charts

Objectives

- Build network and hierachical maps in D3.

Basic template

As usual, let's start with a simple HTML template.

```
<html>
<head>
  <style></style>
</head>
<body>
  <svg></svg>
<script src="https://d3js.org/d3.v6.min.js"></script>
</body>
</html>
```

Add in this D3 code.

```
let width = 800,
    height = 800;

let svg = d3.select("svg")
    .attr("width", width)
    .attr("height", height);

let data = [];
for (let i=0; i < 20; i++) {
    let obj = {};
    data.push(obj);
}

let node = svg.append("g")
  .attr("id", "nodes")
  .selectAll("circle")
  .data(data)
  .enter()
  .append("circle")
    .attr("r", 25)
    .style("fill", "steelblue");
```

We created a bunch of circles, but we need to space them using D3's force simulation.

```
let simulation = d3.forceSimulation()
    .nodes(data)
    .force("x", d3.forceX().strength(0.5).x( width / 2 ))
    .force("y", d3.forceY().strength(0.5).y( height /2 ))
    .force("charge", d3.forceManyBody().strength(20))
    .force("collide", d3.forceCollide().strength(1).radius(30))
    .on("tick", d => {
        node
        .attr("cx", d => d.x)
        .attr("cy", d => d.y);
    });
```

What's with the data array?

What sorcery is this? We didn't add x and y values to the **data** array, so how are the cx and cy attributes updated?

Console.log the **data** array.

Lookup the D3 force documentation.

Force simulation

Play with the values in the force simulation to get an intuitive feel of what the code does.

Can you tweak the amount of cicles, move the center of gravity, create a repelling force, change the collision radius?

Add drag interactivity

```
...
.call(d3.drag()
    .on("start", dragstarted)
    .on("drag", dragged)
    .on("end", dragended));

function dragstarted(event, d) {
  if (!event.active) simulation.alphaTarget(0.3).restart();
  d.fx = d.x;
  d.fy = d.y;
}

function dragged(event, d) {
  d.fx = event.x;
  d.fy = event.y;
}

function dragended(event, d) {
  if (!event.active) simulation.alphaTarget(0);
  d.fx = null;
  d.fy = null;
}
```

With drag added, should be easier to see what some of the parameters do, like d3.forceManyBody.

3 classes of nodes

Let's say our data has 3 classes of nodes. Randomly generating them:

```
for (let i=0; i < 20; i++) {
    let obj = {};
    obj.class = Math.floor(Math.random() * 3);
    data.push(obj);
}
```

Can you give each type its own color?

Different centre position

Let's define a different centre x position for each class of nodes.

```
let xPosition = d3.scaleOrdinal()
  .domain([0, 1, 2])
  .range([150, 400, 650]);

let simulation = d3.forceSimulation()
    .force("x", d3.forceX().strength(0.5).x( d => xPosition(d.class) ))
    .force("y", d3.forceY().strength(0.2).y( height /2 ))
    .force("charge", d3.forceManyBody().strength(25))
    .force("collide", d3.forceCollide().strength(1).radius(30))
```

Can you add in more nodes, and also more groups as well?

Interactivity

```
<button id="group1">Multiple Groups</button>
<button id="group2">One Center</button>
```

```
d3.select("#group1").on("click", function() {
    simulation
    .force("x", d3.forceX().strength(0.5).x(d => xPosition(d.class)))
    .force("y", d3.forceY().strength(0.2).y( height /2 ))
    .alphaTarget(0.3)
    .restart();
})

d3.select("#group2").on("click", function() {
    simulation
    .force("x", d3.forceX().strength(0.1).x(400))
    .force("y", d3.forceY().strength(0.1).y(400))
    .alphaTarget(0.3)
    .restart();
})
```

Add links data

We'll need to give each node an id, and specify a source and target for each link.

```
let data = [];
for (let i=0; i < 20; i++) {
    let obj = {};
    obj.id = "node" + i;
    obj.class = Math.floor(Math.random() * 3);
    data.push(obj);
}

let links = [];
for (let i=0; i < 10; i++) {
    let obj = {};
    obj.source = "node" + Math.floor(Math.random() * 20);
    obj.target = "node" + Math.floor(Math.random() * 20);
    links.push(obj);
}
```

Console.log **data** and **links** arrays to inspect the structure.

Draw the links

We'll add the link paths to the SVG. No path data yet, which we'll update in the simulation on tick function.

```
let linkpath = svg.append("g")
    .attr("id", "links")
    .selectAll("path")
    .data(links)
  .enter()
    .append("path")
    .attr("fill", "none")
    .attr("stroke", "black");
```

Update force simulation

Putting everything together.

```
let simulation = d3.forceSimulation()
    .nodes(data)
    .force("x", d3.forceX().strength(0.5).x( d => xPosition(d.class) ))
    .force("y", d3.forceY().strength(0.2).y( height /2 ))
    .force("link", d3.forceLink(links).id(d => d.id))
    .force("charge", d3.forceManyBody().strength(20))
    .force("collide", d3.forceCollide().strength(1).radius(30))
    .on("tick", d => {
        node
        .attr("cx", d => d.x)
        .attr("cy", d => d.y);

        linkpath
        .attr("d", d => "M" + d.source.x + "," + d.source.y + " " + d.target.x + "," + d.target.y);
    });
```

Console.log **simulation.nodes()** and **simulation.force("link").links()** to inspect the structure.

Link parameters

Links can have distances and strengths. Play around with the properties.

```
...
.force("link", d3.forceLink(links)
    .id(d => d.id)
    .distance(50)
    .strength(0.5)
 )
...
```

Can you change the link path to be a curved path?

Assignment 5

Putting it all together

Setup

To be completed before **29-4-2021**(Thurs) 1200hrs

- This repo link will be of the format https://**<username>**.github.io/HASS-assignment5
- In the forked repo create a basic **index.html** file, and make it visible on GitHub repo pages (it's under the settings tab, under GitHub pages).
- You can put your CSS style tags, HTML and JS script code into one file, or organize them into files/directories.

What: Dataset

- We'll be looking at the recent Budget 2021.
- Alternatively, you can get this budget data via API at the data.gov.sg portal.
- Whether you work in CSV, or JSON, locally or via an API it is up to you.
- Note: You have to do quite a bit of data manipulation for this piece.

Why: Presentation

- Your boss has asked you to present a FY2021 overview of the government expenditure.
- You notice that the structure of the data is such that it is of a **hierarchical nature**.
- E.g. at the highest level it is total expenditure, broken down into sectors, further broken down into ministries, which is further broken down into expenditure type (operating and development).
- You don't have to represent the whole structure - just what you feel is interesting to present.
- You can even use faceting to encode what you think makes sense.

How: Encoding

- Up to you. The dataset is a tree-like structure, which lends itself well to several encodings.
- Sample ones include treemap, sunburst, icicle, packed circle, dendogram, etc.
- You could decide to add interaction or simply present everything as a flat chart.
- You could decide to facet the chart to just show off what you need.
- You could keep it in a tabular form - stacked area, stacked bar - with interaction.

Questions?

Chi-Loong | V/R