Lecture Exercise 8 (5/23)

Submit your team number

Question *Submitted May 23rd 2023 at 5:03:59 pm*

Please enter your team number.

12

1. Add a pulling force from the center of the canvas

Open exercise8.html. Currently you can only see this on the canvas.

You can move the entire graph by translating all nodes and links. “Center” force allows you to add a pulling force, initiated from a designated (x, y) coordinate. Read this section about forceCenter and change your code accordingly.

Your output result should be like:

Question 1 *Submitted May 23rd 2023 at 5:09:06 pm*

Copy and paste your .force() call defining the “center” force.

```
const force = d3.forceSimulation(dataset.nodes)
  .force("charge", d3.forceManyBody())
  .force('center', d3.forceCenter(width / 2, height / 2))
  .force("link", d3.forceLink(dataset.edges))
```

Question 2 *Submitted May 23rd 2023 at 5:10:56 pm*

Remove the “link” force by commenting out `.force("link", d3.forceLink(dataset.edges))` so

that your force simulator only has the “charge” and “center” forces. How does your output change? Why?

Remove the “charge” force instead of “link” as well and explain why output looks as you see.

The output changes by no long having any links. This causes the nodes to expand outward.

The nodes converge to the center. This is because the center is forcing all the nodes in thesame direction

2. Add a pulling force from multiple focal points

You can also group/align data items along a specific horizontal or vertical line. All you need to do is to add a pulling force from a certain point; and you can vary the coordinate of the point by a certain value in the data. For example, the graph below removed edges and placed nodes with the same agerange closer to each other.

Left: nodes whose agerange is 1. Centered around (1*150, height/2)

Right: nodes whose agerange is 2. Centered around (2*150, height/2)

You can do so by:

1. **Removing** the "link" force.
2. Adding forceX and forceY force. Read this section about forceX and forceY of the same webpage and change your code accordingly to get the graph above.

Question *Submitted May 23rd 2023 at 5:19:33 pm*

Copy and paste your new definition of the force method chain

```
var force = d3.forceSimulation(dataset.nodes)
              .force("charge", d3.forceManyBody())
              .force('center', d3.forceCenter(width/2, height/2))
              .force('x', d3.forceX().x(d => d.agerange * 150))
              .force('y', d3.forceY().y(d => height/2))
```

3. Add a "collide" shield

Change the definition of the variable force as below.

```
let force = d3.forceSimulation(dataset.nodes)
    .force('charge', d3.forceManyBody().strength(1))
    .force('center', d3.forceCenter(width/2, height/2));
```

Question 1 *Submitted May 23rd 2023 at 5:22:04 pm*

How does .strength(1) impact the graph? Refer to d3-force documentation. Also feel free to comment it or change the number inside strength()

> It forces the nodes to converge at the center because the direction of the force is defined and it is applying of size 1.

Question 2 *Submitted May 23rd 2023 at 5:24:22 pm*

Notice that all the nodes are now on top of each other.

“Collide” type force ensures that the two nodes do not overlap each other, by defining **a circular shield** around the node with the radius r. Once defined, any other node cannot be placed inside that r radius.

Refer to the "forceCollide" section of this documentation and revise `force` on your code. Copy/paste the revised `force` .

```
let force = d3.forceSimulation(dataset.nodes)
    .force('charge', d3.forceManyBody().strength(1))
    .force('center', d3.forceCenter(width/2, height/2))
    .force('collision', d3.forceCollide().radius(function(d) {
        return d.value
    }))
```

4. Add radio buttons to change the size of each circle

The output seems to work, but you probably noticed that it's quite difficult to read the plot as some circles are very small. To make our force-directed graph more interactive and readable, two radio buttons are added on the html document with two values - default and large. They are expected to change the size of each circle.

```
<div id="radio">
    <input type="radio" id="default" name="size" value="default" checked>
    <label for="default">Default Size</label>
    <input type="radio" id="large" name="size" value="large">
    <label for="large">Large Size</label>
</div>
```

We have already written a listener function for you. When you click on the "Default Size/ Large Size" buttons, the variable `size` will be changed to `default/large` accordingly.

```
const radio = d3.select("#radio").on("change", function(d) {
    const size = d.target.value;
    console.log(size);
    //TODO

});
```

When the value we read from the radio buttons (d.target.value) is "`default`", you do not need to change the value. When it is "`large`", you need to multiply these value by 1.5.

```
nodes: [
    { agerange: 1, name: "Adam"    , value: 4 },
    { agerange: 2, name: "Bob"     , value: 10 },
    { agerange: 2, name: "Carrie"  , value: 7},
    { agerange: 1, name: "Donovan" , value: 1},
    { agerange: 2, name: "Edward"  , value: 13},
    { agerange: 2, name: "Felicity", value: 8},
    { agerange: 2, name: "George"  , value: 5},
    { agerange: 1, name: "Hannah"  , value: 15},
    { agerange: 2, name: "Iris"    , value: 17},
    { agerange: 1, name: "Jerry"   , value: 3}
],
```

Hint: Instead of processing the raw data, you may consider change the `r` attribute in svg `nodes` variable.

Question *Submitted May 23rd 2023 at 5:26:53 pm*

Copy/paste your listener function below.

```
            if (size == "default"){
                  nodes.attr('r', d => d.value);
              } else if (size == "large"){
                  nodes.attr('r', d => d.value * 1.5);
              }
```

5. Beeswarm Plot

Download index-beeswarm.html and beeswarm.js.

In the starter code, we have created `nodes` and `nodeElements` for you.

- `nodes`: This variable is an array that holds the data for each node in the graph. Each node in the array is represented as an object with two properties:
  - `radius`: It determines the radius of the node, randomly generated between 5 and 15 (`Math.random() * 10 + 5`).
  - `xAttribute`: It represents the x-coordinate attribute of the node, randomly generated between -100 and 100 (`Math.random()*200-100`).
- `nodeElements`: This variable is a selection in D3.js that represents the visual elements (`<circles>`) corresponding to the `nodes` data. The `attr("r", (d) => d.radius)` sets the radius of each circle based on the corresponding node's `radius` property.

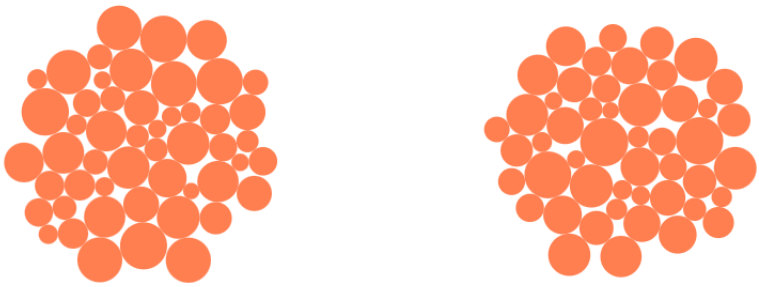
By combining couple types of force, you can draw a beeswarm plot. You need to add the following types of force:

- `d3.forceX().x()`:
  - This force is responsible for positioning the nodes along the x-axis. Based on the value of the `xAttribute` property of each node, if `xAttribute` is less than **0.5**, the node is placed on the left side by returning `xScale(-50)`. If not, the node is placed on the right side by returning `xScale(50)`
- `d3.forceY(height / 2)`:
  - This force is responsible for vertically centering the nodes along the y-axis. Setting the input parameter to be `(height/2)`. So that the force is applied to each node individually and sets their y-coordinate to half of the graph's height `(height/2)`, effectively centering them vertically.
- `d3.forceCollide().radius(d => d.radius)`

Finally, inside the `simulation.on()` method:

- Updates the `nodeElements` attributes `cx` and `cy` of each circle element using the `d.x` and `d.y` values from the simulation's nodes.

Expected output should look as below:



Question *Submitted May 23rd 2023 at 5:33:11 pm*

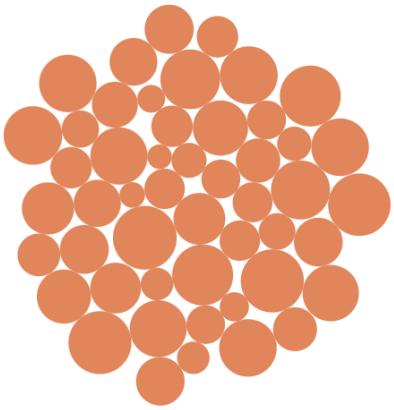
Copy/paste your definition of 1) `simulation` and 2) what needs to happen for each tick.

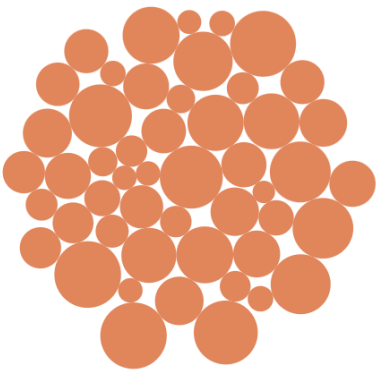
```
simulation = d3.forceSimulation(nodes)
  .force("x", d3.forceX().x( (d) => (d.xAttribute < 0.5) ? xScale(-50) : xScale(50)))
  .force("y", d3.forceY(height / 2))
  .force("collide", d3.forceCollide().radius(d=> d.radius));
simulation.on("tick", () => {
    nodeElements.attr("cx", (d) => d.x).attr("cy", (d) => d.y);
});
```

Upload Your Files

Question 1 *Submitted May 23rd 2023 at 5:33:36 pm*

Upload the screenshot of your resulting webpage. You will need to click the "clip" button to upload a file into the Answer box.





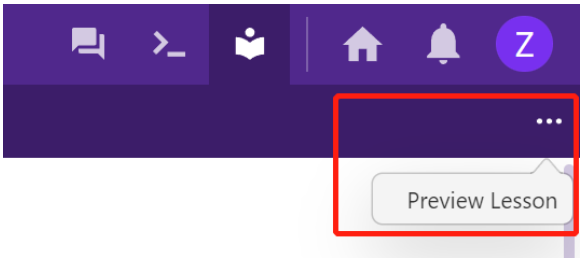
Displaying exercise8.html

◉ Default Size ○ Large Size

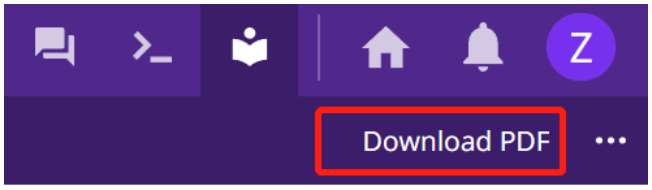
Question 2 *Submitted May 23rd 2023 at 5:33:41 pm*

You need to download the PDF of lecture exercise 7 and upload it with other files to the Gradescope. Follow the instructions on how to download PDF file:

1. Click on the ellipsis button and the Preview Lesson.



2. After that, click on the Download PDF button.



- [x] PDF downloaded!
- [ ] Haven't done yet!

Question 3 *Submitted May 23rd 2023 at 5:33:43 pm*

Upload the following files to Gradescope. You need to make **a group submission,adding all present members in your team**, so that the present members get the participation credit.

Files to upload:

- exercise7.html
- PDF you downloaded as Q2

- [x] Our team uploaded the the files on gradescope!
- [ ] Oops, our team did not upload the files on gradescope!

Feedback

Question *Submitted May 23rd 2023 at 5:33:47 pm*

Was the activity today clear? If not, please share how the course can improve it. Your comments will help us design future lab content (and also future students).

No response