const data = {

"name": "flare",

"size": 20000,

"label": "test",

"children": [

{

"name": "North America",

"children": [

{"C1": "DragForce", "size": 10820},

{"C2": "GravityForce", "size": 13360},

{"C3": "IForce", "size": 3190},

]

},

{

"name": "South America",

"children": [

{"C3": "DragForce", "size": 10820},

{"C4": "GravityForce", "size": 13360},

{"C5": "IForce", "size": 3190},

]

},

{

"name": "Africa",

"children": [

{"C6": "DragForce", "size": 10820},

{"C7": "GravityForce", "size": 13360},

{"C8": "IForce", "size": 3190},

]

},

{

"name": "Asia",

"children": [

{"C9": "DragForce", "size": 10820},

{"C10": "GravityForce", "size": 13360},

{"C11": "IForce", "size": 3190},

]

},

{

"name": "Europe",

"children": [

{"C12": "DragForce", "size": 10820},

{"C13": "GravityForce", "size": 13360},

{"C14": "IForce", "size": 3190},

]

},

{

"name": "Australia",

"children": [

{"C15": "DragForce", "size": 10820},

{"C16": "GravityForce", "size": 13360},

{"C17": "IForce", "size": 3190},

]

}

]

}

const width = 10,

height = 10;

let i = 0;

const root = d3.hierarchy(data);

const transform = d3.zoomIdentity;

let node, link;

const svg = d3.select('#scatter').append('svg')

.call(d3.zoom().scaleExtent([1/4, 8]).on('zoom', zoomed))

.append('g')

.attr('transform', 'translate(300, 125)');

const simulation = d3.forceSimulation()

.force('link', d3.forceLink().id(function(d) { return d.id; }))

.force('charge', d3.forceManyBody().strength(-50).distanceMax(300))

.force('center', d3.forceCenter( width/2, height/4 ))

.on('tick', ticked)

function update() {

const nodes = flatten(root)

const links = root.links()

link = svg

.selectAll('.link')

.data(links, function(d){ return d.target.id })

link.exit().remove()

const linkEnter = link

.enter()

.append('line')

.attr('class', 'link')

.style('stroke', '#51a1dc' )

.style('opacity', '0.2')

.style('stroke-width', 2)

link = linkEnter.merge(link)

node = svg

.selectAll('.node')

.data(nodes, function(d){ return d.id })

node.exit().remove()

const nodeEnter = node

.enter()

.append('g')

.attr('class', 'node')

.attr('stroke', '#666')

.attr('stroke-width', 2)

.style('fill', color)

.style('opacity', 1)

.on('click', clicked)

.call(d3.drag()

.on('start', dragstarted)

.on('drag', dragged)

.on('end', dragended))

nodeEnter.append('circle')

.attr("r", function(d) { return Math.sqrt(d.data.size) / 10 || 4.5; })

.style('text-anchor', function(d){ return d.children ? 'end' : 'start'; })

.text(function(d){ return d.data.name })

node = nodeEnter.merge(node)

simulation.nodes(nodes)

simulation.force('link').links(links)

}

function sizeContain(num) {

num = num > 1000 ? num/1000 : num/100

if (num < 4) num = 4

return num

}

function color(d) {

return d.\_children ? "#51A1DC" // collapsed package

: d.children ? "#51A1DC" // expanded package

: "#F94B4C"; // leaf node

}

function radius(d) {

return d.\_children ? 8

: d.children ? 8

: 4

}

function ticked() {

link

.attr('x1', function(d){ return d.source.x; })

.attr('y1', function(d){ return d.source.y; })

.attr('x2', function(d){ return d.target.x; })

.attr('y2', function(d){ return d.target.y; })

node

.attr('transform', function(d){ return `translate(${d.x}, ${d.y})`})

}

function clicked(d) {

if (!d3.event.defaultPrevented) {

if (d.children) {

d.\_children = d.children;

d.children = null;

} else {

d.children = d.\_children;

d.\_children = null;

}

update()

}

}

function dragstarted(d) {

if (!d3.event.active) simulation.alphaTarget(0.3).restart()

d.fx = d.x

d.fy = d.y

}

function dragged(d) {

d.fx = d3.event.x

d.fy = d3.event.y

}

function dragended(d) {

if (!d3.event.active) simulation.alphaTarget(0)

d.fx = null

d.fy = null

}

function flatten(root) {

const nodes = []

function recurse(node) {

if (node.children) node.children.forEach(recurse)

if (!node.id) node.id = ++i;

else ++i;

nodes.push(node)

}

recurse(root)

return nodes

}

function zoomed() {

svg.attr('transform', d3.event.transform)

}

update()