CytoscapeJs vs D3.js

Workshop Work4Graph - 7 décembre 2020

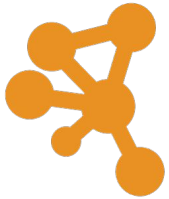
Retour d'expérience & tutoriels

Marie Lefebvre
UMR 1332 Biologie du Fruit et Pathologie,
Bordeaux

CytoscapeJS

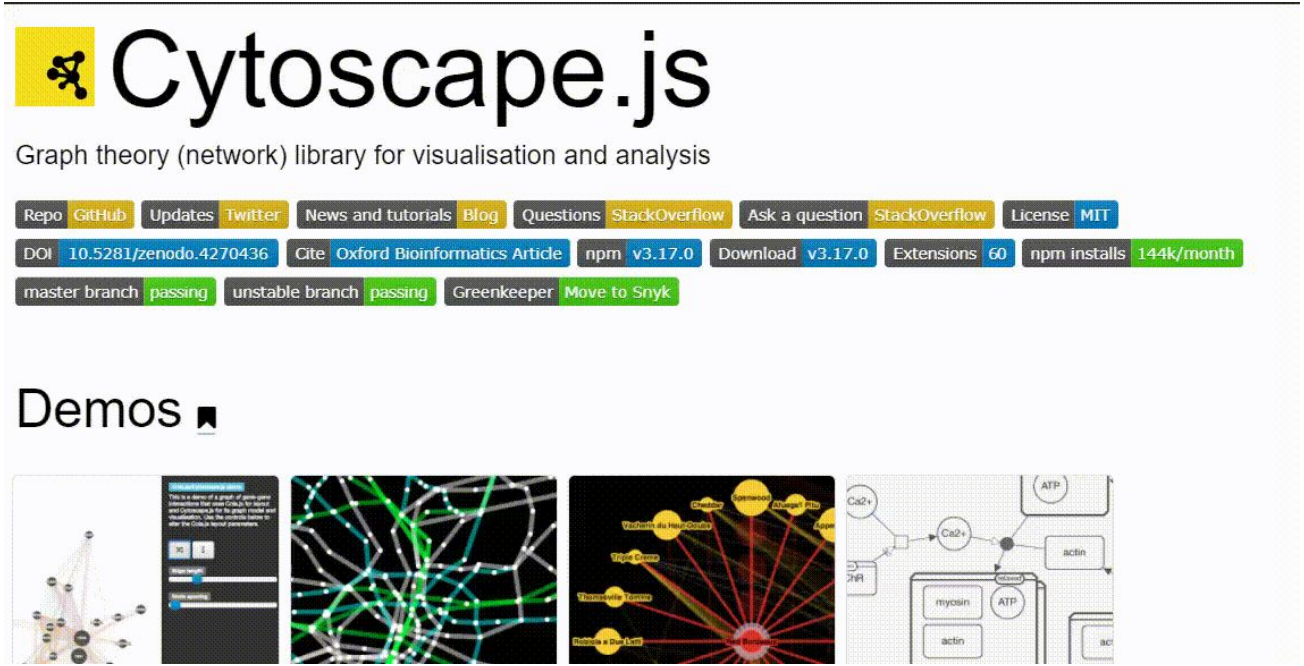
CytoscapeJS en bref

- Librairie JavaScript pour la visualisation interactive de **réseau** via Canvas
- Développé par le Cytoscape Consortium
- Installation via npm, bower, yarn ou en sourçant cytoscape.min.js
- Documentation: https://js.cytoscape.org/



Max Franz et al., 2016

CytoscapeJS la documentation



Les promesses

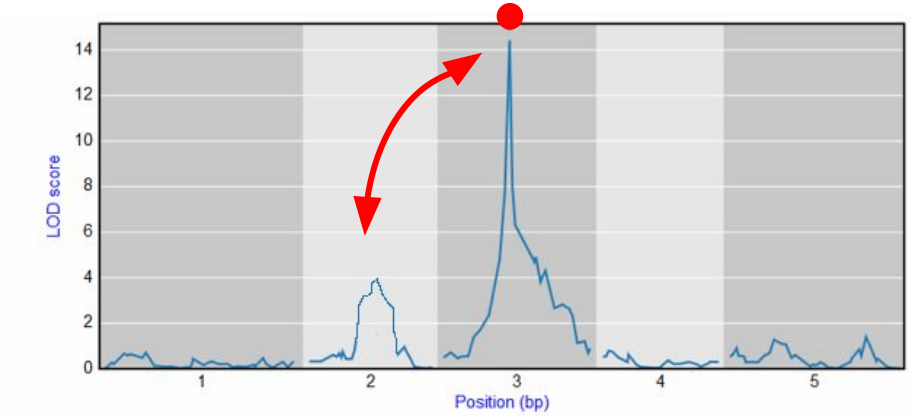
- Facile à implémenter en peu de lignes de code grâce à une API
  - **core**: point d'entrée, modification du style, ajout d'éléments, fonctions
  - **collection**: une API pour filtrer, parcourir, effectuer des opérations
- Types de graphes & layout : traditionnels, dirigés, non dirigés, les multigraphes et les hypergraphes (avec des nœuds composés, mais pas encore avec des hypergraphes).

Les promesses

- Facile à implémenter en peu de lignes de code grâce à une API
  - **core**: point d'entrée, modification du style, ajout d'éléments, fonctions
  - **collection**: une API pour filtrer, parcourir, effectuer des opérations
- Types de graphes & layout : traditionnels, dirigés, non dirigés, les multigraphes et les hypergraphes (avec des nœuds composés, mais pas encore avec des hypergraphes).
- Reproduction de plusieurs fonctionnalités de Cytoscape:
  - Style graphique similaire
  - Fonctions d'analyse de réseau (mesure de centralité, recherche de connectivité, chemin le plus court,...)
- Export d'un réseau Cytoscape vers CytoscapeJS

Retour d'expérience

- Etude des voix de régulation de la germination chez des graines d'*Arabidopsis thaliana*
- Via analyse QTL ⇒ identification de hotspots
- **Problème**: un QTL regroupe plusieurs gènes, quel est le gène causal ?



- ❖ AraQTL
- ❖ Base web semantique

Harm Nijveen
WUR

Retour d'expérience

- Représentation de réseaux régulation
  - implémentation rapide
  - graphe dirigé
  - interactivité moyenne

Résultat obtenu

- ~1 semaine de développement
- 150 lignes js

D3.js

D3.js en bref

- D3 est l'acronyme de Data-Driven Documents
- Librairie JavaScript open-source développée par Mike Bostock
- Visualisations de données interactives via HTML, CSS et **SVG**
- Documentation https://github.com/d3/d3/wiki & https://www.d3-graph-gallery.com
- Installation via npm, cdn ou en sourçant d3js.min.js

Les promesses

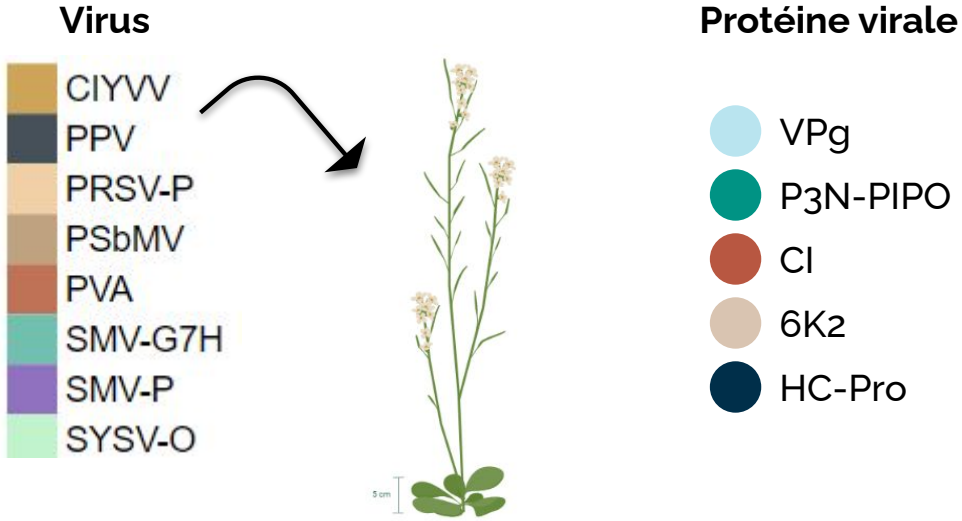
- Types de graphes & layout : non-dirigé et non-pondérée, dirigé et non-pondéré, non-dirigé et pondéré, dirigé et pondéré
- Personnalisation: inclure d'autres représentations (histogramme, pie chart,...)

Retour d'expérience

Sylvie German Retana

Guillaume Lafforgue

- Etude des interactions entre protéines virales et protéines de l'hôte (*Arabidopsis*)
  - ➜ contribution au processus d'infection

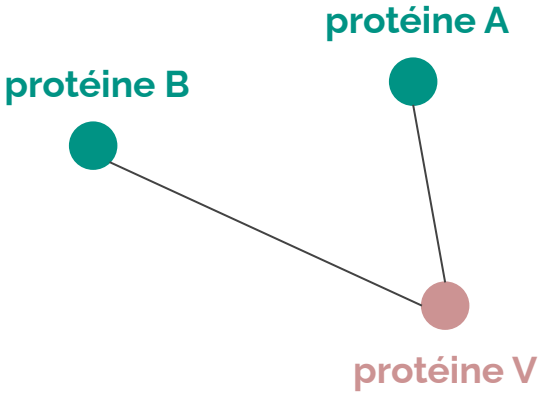


Retour d'expérience

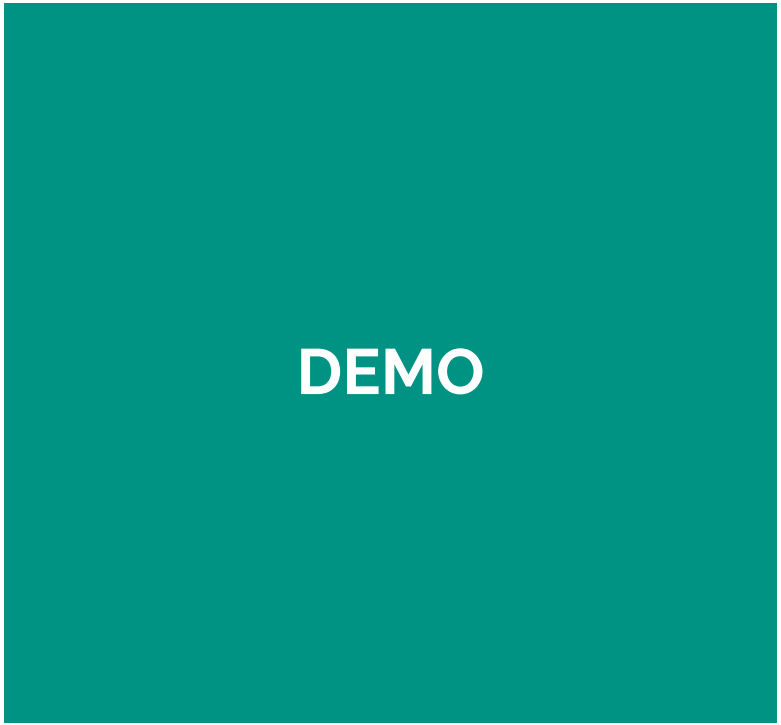
Sylvie German Retana

Guillaume Lafforgue

- Etude des interactions entre protéines virales et protéines de l'hôte (*Arabidopsis*)
  - ➜ contribution au processus d'infection
- Représentation des PPIs
  - graphe de connexions
  - Très customisable
  - Interactivité élevée



Résultat obtenu



- ~1 mois de développement
- 380 lignes js

CytoscapeJS vs D3.js

CytoscapeJS

- Documentation riche mais qui manque parfois de détail
- API facile à utiliser
- Faire beaucoup avec moins de code
- Personnalisation limitée
- Des extensions pas toujours à jour

D3.js

- Documentation riche
- Large communauté
- Personnalisation presque infinie
- Les premiers pas un peu difficiles
- Code peut vite devenir illisible
- Beaucoup de documentation deprecated

Mini tuto

</>

CytoscapeJS

CytoscapeJS HTML & CSS

HTML

```
<head>
  <script src="cytoscape.min.js"></script>
  <link rel="stylesheet" href="cy.css"></link>
</head>
<body>
  <div id="cy"></div>
  <script src="network-cy.js"></script>
</body>
```

CSS

```
#cy {
  width: 100%;
  height: 100%;
  position: absolute;
  top: 0px;
  left: 0px;
}
```

CytoscapeJS la base

JS

```
// initialize cytoscape object
var cy = cytoscape({
   container: document.getElementById('cy'),     // container

});
```

CytoscapeJS la base

JS

```
// initialize cytoscape object
var cy = cytoscape({
   container: document.getElementById('cy'),     // container
   elements: {
      nodes: [
         { data: { id: 'a' } },
         {  data: { id: 'b' } },
         { data: { id: 'c' } }
      ],
      edges: [
       { data: { id: 'ab', source: 'a', target: 'b' } },
       { data: { id: 'bc', source: 'b', target: 'c' } }
      ]
    },
});
```

CytoscapeJS la base

JS

```
// initialize cytoscape object
var cy = cytoscape({
   container: document.getElementById('cy'),     // container
   elements: {
      nodes: [
         { data: { id: 'a' } },
         {  data: { id: 'b' } },
         { data: { id: 'c' } }
      ],
      edges: [
       { data: { id: 'ab', source: 'a', target: 'b' } },
       { data: { id: 'bc', source: 'b', target: 'c' } }
      ]
    },
    style: [                              // stylesheet
     {
        selector: 'node',
        style: {
           'shape': 'circle',
           'background-color': 'red'
        }
     }]
});
```

CytoscapeJS la base ++

JS

```
// initialize cytoscape object
var cy = cytoscape({
   container: document.getElementById('cy'),     // container
   elements: {
      nodes: [
         { data: { id: 'a' } },
         {  data: { id: 'b' } },
         { data: { id: 'c' } }
      ],
      edges: [
       { data: { id: 'ab', source: 'a', target: 'b' } },
       { data: { id: 'bc', source: 'b', target: 'c' } }
      ]
    },
    style: [                          // stylesheet
     {
        selector: 'node',
        style: {
           'shape': 'circle',
           'background-color': 'red',
          'label': 'data(id)'
        }
      }]
});
```

Mini tuto

</>

D3js

D3.js HTML

HTML

```
<meta charset="utf-8">
<head>
  <script src="d3.min.js"></script>
</head>
<body>
  <div id="viz"></div>
  <script src="network-d3.js"></script>
</body>
```

D3.js la base

JS

```
// set the dimensions and margins of the graph
var margin = {top: 10, right: 10, bottom: 10, left: 10},
    width = 800 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;


// append the svg object to the body of the page
var svg = d3.select("#viz")
     .append("svg")
     .attr("width", width + margin.left + margin.right)
     .attr("height", height + margin.top + margin.bottom)
```

D3.js la base

JS

```
// Load data
var data = {
   "nodes": [
      {  "id": "A" },
      { "id": "B" },
      {  "id": "C" }
   ],
   "links": [
      {  "source": "A",  "target": "B"  },
      { "source": "B", "target": "C" }
   ]
};
```

D3.js la base

JS

```
// Initialize the links
var link = svg
   .selectAll("line")
   .data(data.links)
      .enter()
         .append("line")
         .style("stroke", "black")

// Initialize the nodes
var node = svg.append("g")
   .attr("class", "nodes")
   .selectAll("g")
   .data(data.nodes)
      .enter()
         .append("g")
```

D3.js la base

JS

```
// Bind circle to node
var circles = node.append("circle")
    .attr("r", 15)
    .style("fill", "red")
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; });
```

D3.js la base

JS

```
// Let's list the force we wanna apply on the network
// Force algorithm is applied to data.nodes
var simulation = d3.forceSimulation(data.nodes)
        .force("link", d3.forceLink()           // This force provides links between nodes
            .id(function(d) { return d.id; })   // This provide  the id of a node
            .links(data.links)                  // and this the list of links
        )
        // This adds repulsion between nodes
        .force("charge", d3.forceManyBody().strength(-400))
        // This force attracts nodes to the center of the svg area
        .force("center", d3.forceCenter(width / 2, height / 2))
        .on("end", ticked);
```

D3.js la base

JS

```
// This function is run at each iteration of the force algorithm, updating the nodes position.
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 + ")";
        });
}
```

D3.js la base ++

JS

```
// Bind node label to node
node.append("text")
    .attr("dx", -5)
    .attr("dy", 5)
    .text(function(d) { return d.id; });
```

D3.js la base +++

JS

```
// Bind circle to node
var circles = node.append("circle")
    .attr("r", 15)
    .style("fill", "red")
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; })
    .call(d3.drag().on("drag", dragged));


function dragged(event, d) {
    d.x = event.x, d.y = event.y;
    d3.select(this.parentNode)
        .attr("transform", function(d) {
            return "translate(" + d.x + "," + d.y + ")";
        });
    link.filter(function(l) { return l.source === d; }).attr("x1", d.x).attr("y1", d.y);
    link.filter(function(l) { return l.target === d; }).attr("x2", d.x).attr("y2", d.y);
}
```

Conclusion

- CytoscapeJS : 34 lignes **VS** d3.js: 99 lignes

- Interaction proteine-proteine virale: https://github.com/marieBvr/PVI
- QTL chez Arabidopsis: http://www.bioinformatics.nl/AraQTL
- Tutoriels: https://github.com/marieBvr/tutorials-work4graph

Merci !