



Université de lorraine, UFR-MIM, M2 – IHM

1

Sommaire

- **Partie 1**

- Introduction
- Fonctionnalités
- Diagrammes

- **Partie 2**

- Transitions
- Interactions
- Applications

Partie 1

D3.js c'est quoi ? (D3 pour Data-Driven Documents)

- Une librairie JavaScript libre
- Destinée à générer des documents graphiques à partir des données
- Utilisée pour la représentation visuelle de données
 - Ex : catégorielles, numériques, dynamiques, relationnelles, hiérarchiques, géographiques...
- Successeur de *Protovis*
 - *Popularisée en 2011 et développée par Michael Bostock, Vadim Ogievetsky et Jeffrey Heer*
- Utilise HTML, CSS et SVG

Pourquoi D3.js ?

- Compatible avec la majorité des navigateurs web et conforme aux normes W3C
- Permet d'avoir :
 - des interactions efficaces avec l'utilisateur grâce à son gestionnaire d'évènements
 - des transitions fluides entre les différents états de l'interface utilisateur
- Dispose d'une panoplie d'API réutilisables :
 - [Arrays](#), [Axes](#), [Brushes](#), [Chords](#), [Collections](#), [Colors](#), [Color Schemes](#), [Contours](#), [Dispatches](#), [Dragging](#), [Delimiter-Separated Values](#), [Easings](#), [Fetches](#), [Forces](#), [Number Formats](#), [Geographies](#), [Hierarchies](#), [Interpolators](#), [Paths](#), [Polygons](#), [Quadrees](#), [Random Numbers](#), [Scales](#), [Selections](#), [Shapes](#), [Time Formats](#), [Time Intervals](#), [Timers](#), [Transitions](#), [Voronoi Diagrams](#), [Zooming](#)...etc

Installation

- Directement, en ajoutant le lien de téléchargement dans le HTML

```
<script src="https://d3js.org/d3.v4.min.js"></script>
```

- Ou, en utilisant npm via la commande suivante :

```
npm install d3
```

- Le fichier script nommé *nomFichier.js* sera référencé dans le HTML

```
<script src=" myChart.js "></script>
```

Sélection et manipulation des DOM (1)

- Sans D3.js

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
  var paragraph = paragraphs.item(i);
  paragraph.style.setProperty("color", "green", null);
}
```

- Avec D3.js :

1. ***d3.selectAll()***

```
d3.selectAll("p").style("color", "green");
```

2. ***d3.select()***

```
d3.select('h1').style('color', 'red')
  .attr('class', 'heading')
  .text('Titre h1 modifié');

d3.select('body').append('p').text('Premier paragraphe');
d3.select('body').append('p').text('Deuxième paragraphe');
```

```
d3.select('item')
```

```
d3.select('.itemClass')
```

```
d3.select('#itemId')
```

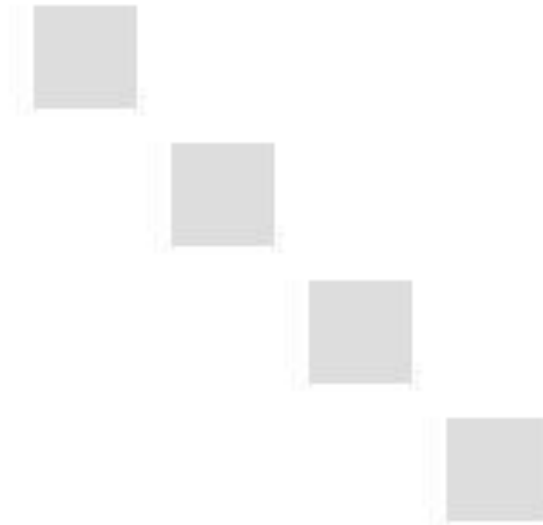
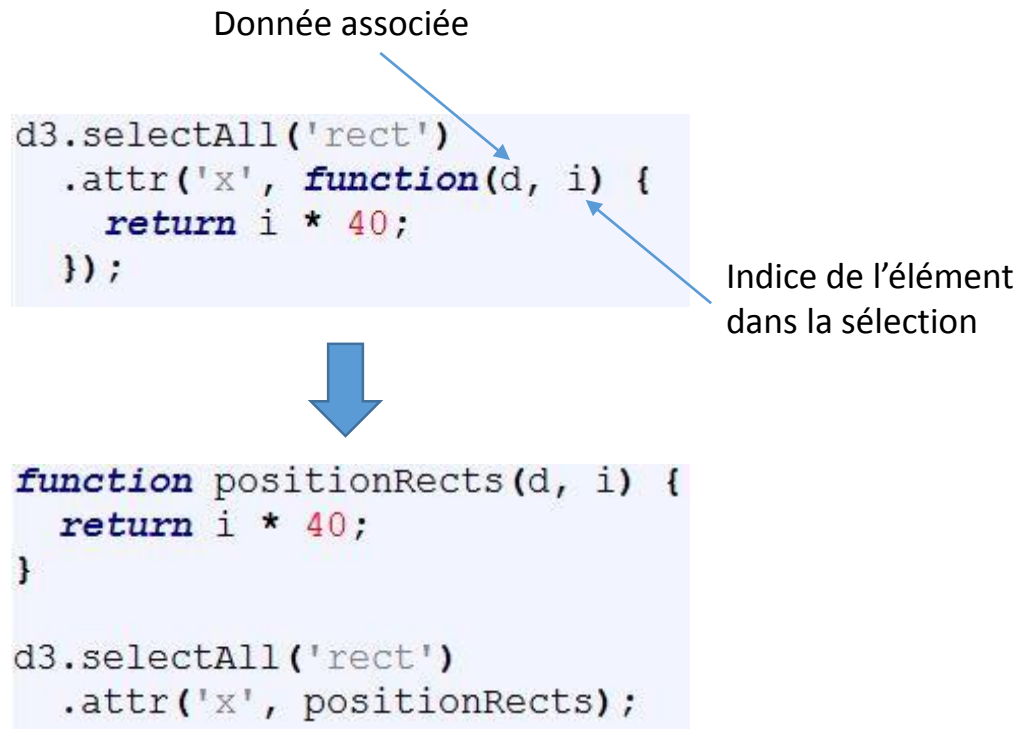
Sélection et manipulation des DOM (2)

- Utilisation des fonctions prédéfinies

Nom	Fonction	Exemple
<code>.style</code>	Modifie le style	<code>d3.selectAll('circle').style('fill', 'red')</code>
<code>.attr</code>	Modifie un attribut	<code>d3.selectAll('rect').attr('width', 10)</code>
<code>.classed</code>	Ajout/suppression un attribut de classe	<code>d3.select('.item').classed('selected', true)</code>
<code>.property</code>	Modifie la propriété d'un élément	<code>d3.selectAll('.checkbox').property('checked', false)</code>
<code>.text</code>	Modifie le contenu texte	<code>d3.select('div.title').text('Mon nouveau cours D3')</code>
<code>.html</code>	Change le contenu HTML	<code>d3.select('.legend').html('<div class="block"></div><div>0 - 10</div>')</code>

Sélection et manipulation des DOM (3)

- Utilisation des fonctions de callback

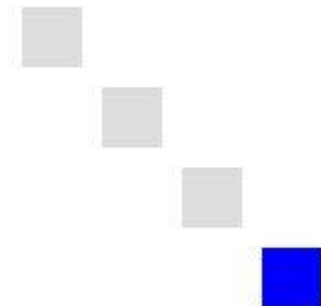


Sélection et manipulation des DOM (4)

- Gestion des évènements d'une sélection

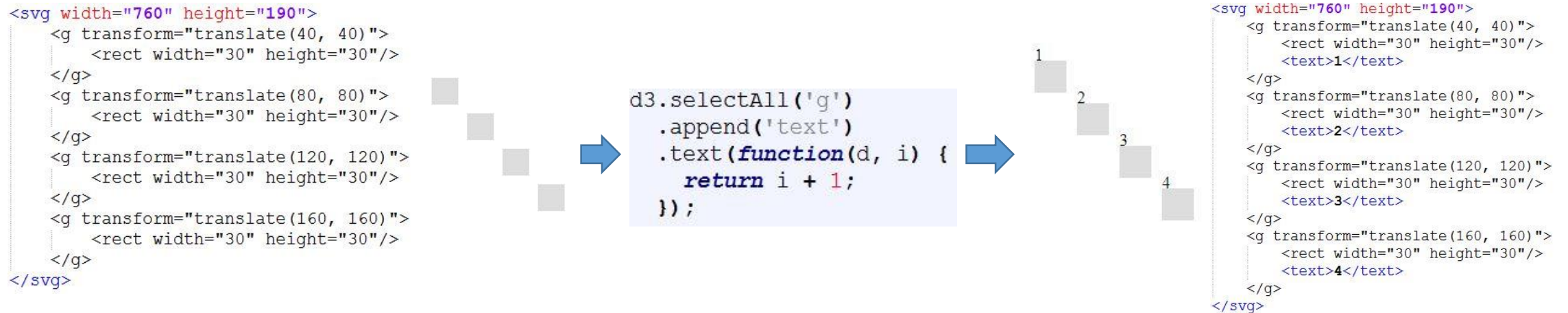
Nom	Description
click	Clique sur l'élément
mouseenter	Entrer du pointeur de la souri dans l'élément
mouseover	Entrer du pointeur de la souri dans l'élément (enfants)
mouseleave	Sortie du pointeur de la souri de l'élément
mouseout	Sortie du pointeur de la souri de l'élément (enfants)
mousemove	Survole du pointeur de la souri sur l'élément

```
d3.selectAll('rect')  
  .on('click', function(d, i) {  
    d3.select(this)  
      .style('fill', 'blue');  
  });
```

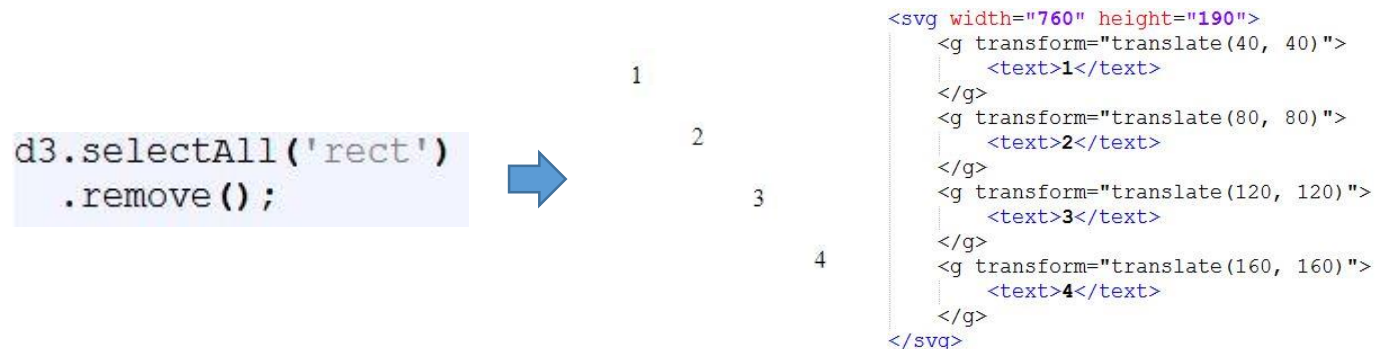


Sélection et manipulation des DOM (5)

- Ajout d'éléments dans une sélection



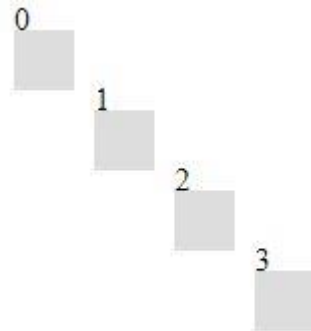
- Suppression des éléments d'une sélection



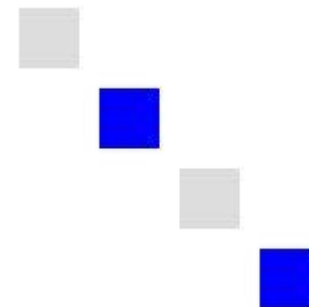
Sélection et manipulation des DOM (6)

- Parcours et filtrage dans une sélection

```
d3.selectAll('g')  
  .each(function(d, i) {  
    d3.select(this)  
      .append('text')  
      .text(i);  
  });
```



```
d3.selectAll('rect')  
  .filter(function(d, i) {  
    return i % 2 === 1;  
  })  
  .style('fill', 'blue');
```

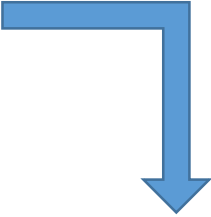


Import des données

- Importer des données à partir d'un fichier CSV


```
Texte,Twittos,Partage  
tweet 1,Toto 1,12  
tweet 2,Toto 2,43  
tweet 3,Toto 3,35  
tweet 4,Toto 4,25
```

tweets.csv



```
d3.csv('tweets.csv', function(err, data) {  
  // Utiliser data  
})
```

Les noms de colonnes du fichier CSV sont utilisés comme noms de propriétés des objets du tableau



```
[  
  {  
    "Texte": "tweet 1",  
    "Twittos": "Toto 1",  
    "Partage": "12"  
  },  
  {  
    "Texte": "tweet 2",  
    "Twittos": "Toto 2",  
    "Partage": "43"  
  },  
  {  
    "Texte": "tweet 3",  
    "Twittos": "Toto 3",  
    "Partage": "35"  
  },  
  {  
    "Texte": "tweet 4",  
    "Twittos": "Toto 4",  
    "Partage": "25"  
  }  
]
```

data

Association des données (1)

- On peut associer à un ensemble d'éléments d'une sélection une collection de données (tableau d'entiers, tableau d'objets...) et delà modifier les éléments DOM sous-jacents en utilisant les données associées.

```
<svg width="760" height="190">
  <g>
    <circle r="30" transform="translate(100,50)" />
    <circle r="30" transform="translate(200,50)" />
    <circle r="30" transform="translate(300,50)" />
    <circle r="30" transform="translate(400,50)" />
  </g>
</svg>
```



```
var tweets = [
  {
    "Texte": "tweet 1",
    "Twittos": "Toto 1",
    "Partage": "12"
  },
  {
    "Texte": "tweet 2",
    "Twittos": "Toto 2",
    "Partage": "43"
  },
  {
    "Texte": "tweet 3",
    "Twittos": "Toto 3",
    "Partage": "35"
  },
  {
    "Texte": "tweet 4",
    "Twittos": "Toto 4",
    "Partage": "25"
  }
];

d3.selectAll('circle')
  .data(tweets)
  .attr('r', function(d) {
    return d.Partage;
  });
```



Association des données (2)

- Que se passera-t-il si la longueur du tableau de données ne correspond pas exactement au nombre d'éléments dans la sélection ?

1. Il y'a plus de données que d'éléments dans la sélection (rajouter des éléments DOM) :

```
<svg width="760" height="190">
  <g>
    <circle r="30" transform="translate(100, 100)"/>
    <circle r="30" transform="translate(200, 100)"/>
  </g>
</svg>
```



```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}
];

// éléments existants
var s = d3.select('g')
  .selectAll('circle')
  .data(tweets)
  .attr('r', function(d) {
    return d.Partage;
  });

// éléments entrants
s.enter().append('circle')
  .attr('transform', function(d, i) {
    return "translate(" + (i+1) * 100 + ", 100)";
  })
  .attr('r', function(d) {
    return d.Partage;
  });
```



```
<svg width="760" height="190">
  <g>
    <circle r="12" transform="translate(100, 100)"/>
    <circle r="43" transform="translate(200, 100)"/>
    <circle r="35" transform="translate(300, 100)"/>
    <circle r="25" transform="translate(400, 100)"/>
  </g>
</svg>
```


Association des données (2)

- Remarques :

- a) Une pratique assez répandue consiste à associer un tableau de données à une sélection vide pour la création d'éléments DOM

```
d3.selectAll('p')  
  .data(tweets)  
  .enter()  
  .append('p')  
  .text(function(d) {  
    return d.Texte;  
  });
```

- b) Il existe une méthode de D3 pour tenir compte à la fois des éléments DOM existants et entrants : `.merge()`

```
// éléments existants  
var s = d3.select('g')  
  .selectAll('circle')  
  .data(tweets);  
  
// éléments entrants et existants  
s.enter().append('circle')  
  .merge(s)  
  .attr('transform', function(d, i) {  
    return "translate(" + (i+1) * 100 + ", 100)";  
  })  
  .attr('r', function(d) {  
    return d.Partage;  
  });
```


Association des données (3)

- Il y'a moins de données que d'éléments dans la sélection (supprimer des éléments DOM) :

```
<svg width="760" height="190">
  <g>
    <circle r="30" transform="translate(100, 100)"/>
    <circle r="30" transform="translate(200, 100)"/>
    <circle r="30" transform="translate(300, 100)"/>
    <circle r="30" transform="translate(400, 100)"/>
  </g>
</svg>
```



```
var tweets = [
  { "Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12" }
];

// éléments sortants
var s = d3.select('g')
  .selectAll('circle')
  .data(tweets)
  .exit()
  .remove();
```



```
<svg width="760" height="190">
  <g>
    <circle r="30" transform="translate(100, 100)"/>
  </g>
</svg>
```



Association des données (4)

- Une bonne pratique consiste à séparer les trois actions (mis à jour, ajout et suppression d'éléments DOM) dans une seule fonction :

```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}
];

function update(data) {

  var s = d3.select('g')
    .selectAll('circle')
    .data(data);

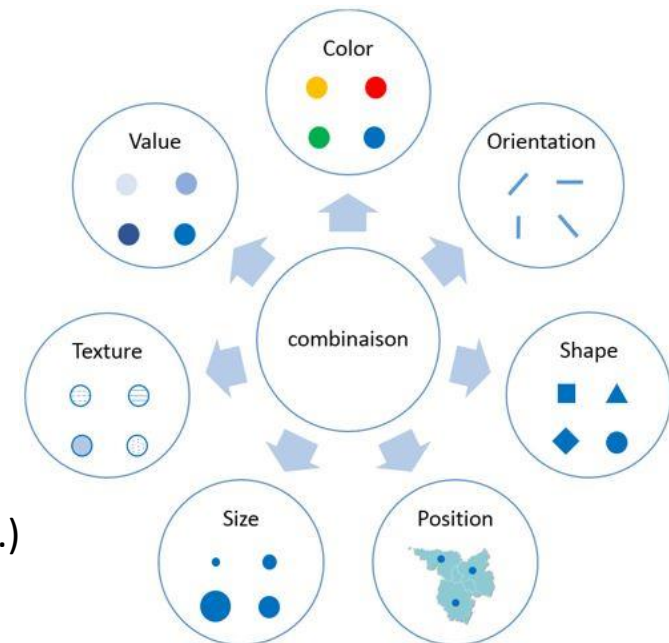
  s.enter()
    .append('circle')
    .merge(s)
    .attr('transform', function(d, i) {
      return "translate(" + (i+1) * 100 + ", 100)";
    })
    .attr('r', function(d) {
      return d.Partage;
    });

  s.exit().remove();
}

update(tweets)
```

Passage à l'échelle (1)

- L'objectif des fonctions de passage à l'échelle est de transposer une donnée en entrée (comprise dans un intervalle prédéfini appelé domaine) vers un intervalle de sortie (appelé rang)
- Généralement les fonctions de passage à l'échelle sont utilisées pour représenter à l'aide d'une variable visuelle une donnée en entrée
 - **Entrée** (*continue* ou *discrète*) : un entier, un réel, une catégorie...etc.
 - **Sortie** (*continue* ou *discrète*) : une variable visuelle
- Exemples :
 - Des dates sur un axe
 - La longueur des bars d'un histogramme
 - Le risque financier via un code couleur (rouge, orange et vert)
 - Une position sur une carte géographique à l'aide d'une catégorie (FR, DE...)
 - ...



Variables visuelles de J. Bertin

Passage à l'échelle (2)

1. Échelle linéaire : Par exemple, pour transposer tweets.Partage

```
var echelleLineaire = d3.scaleLinear()
  .domain([0, 43])
  .range([0, 200]);

echelleLineaire(0); // renvoie 0
echelleLineaire(12); // renvoie 55.813
echelleLineaire(25); // renvoie 116.279
echelleLineaire(35); // renvoie 162.79
echelleLineaire(43); // renvoie 200
```

2. Échelle linéaire domaines multiples : Par exemple, pour transposer tweets.Avis

```
var echelleLineaireMulti = d3.scaleLinear()
  .domain([-1, 0, +1])
  .range(['red', 'yellow', 'green']);

echelleLineaireMulti(-1); // renvoie rgb(255, 0, 0)
echelleLineaireMulti(-0.5); // renvoie rgb(255, 128, 0)
echelleLineaireMulti(0); // renvoie rgb(255, 255, 0)
echelleLineaireMulti(+0.5); // renvoie rgb(128, 192, 0)
echelleLineaireMulti(+1); // renvoie rgb(0, 128, 0)
```



Passage à l'échelle (3)

3. Échelle séquentielle : Par exemple, pour transposer tweets.Like

```
var echelleSequentielle = d3.scaleSequential()  
  .domain([0, 50])  
  .interpolator(d3.interpolateRainbow);
```



Remarque : voir **d3-scale-chromatic** pour plus de fonctions d'interpolation

4. Échelle temporelle : Par exemple, pour transposer tweets.Date

```
var echelleTemporelle = d3.scaleTime()  
  .domain([new Date(2019, 1, 1), new Date(2019, 5, 1)])  
  .range([0, 200]);  
  
echelleTemporelle(new Date(2019, 1, 1)); // renvoie 0  
echelleTemporelle(new Date(2019, 2, 1)); // renvoie 46.682  
echelleTemporelle(new Date(2019, 3, 1)); // renvoie 98.298  
echelleTemporelle(new Date(2019, 4, 1)); // renvoie 148.315  
echelleTemporelle(new Date(2019, 5, 1)); // renvoie 200
```


Passage à l'échelle (4)

5. Échelle de quantité discrète : Par exemple, pour transposer tweets.Popularité

```
var echelleQuantiteDiscrete = d3.scaleQuantize()  
  .domain([0, 50])  
  .range(['pink', 'orange', 'purple', 'bleu', 'brown']);  
  
echelleQuantiteDiscrete(5); // renvoie pink  
echelleQuantiteDiscrete(15); // renvoie orange  
echelleQuantiteDiscrete(25); // renvoie purple  
echelleQuantiteDiscrete(35); // renvoie bleu  
echelleQuantiteDiscrete(45); // renvoie brown
```

6. Échelle ordinale : Par exemple, pour transposer tweets.Groupe

```
var echelleOrdinale = d3.scaleOrdinal()  
  .domain([1, 2, 3, 4, 5, 6])  
  .range(['green', 'bleu', 'pink', 'brown']);  
  
echelleOrdinale(1); // renvoie green  
echelleOrdinale(2); // renvoie bleu  
echelleOrdinale(3); // renvoie pink  
echelleOrdinale(4); // renvoie brown  
echelleOrdinale(5); // renvoie green  
echelleOrdinale(6); // renvoie bleu
```

- Remarque : s'il y a moins d'éléments dans le rang que dans le domaine → le rang se répète

Couleurs

- Un environnement pour manipuler des couleurs : ***d3.color(identifiant)*** → RGB, HEX ou couleur...

```
var rouge = d3.color("red"); // {r: 255, g: 0, b: 0, opacity: 1}
```



- Changement de l'opacité d'une couleur

```
rouge.opacity = 0.7; // {r: 255, g: 0, b: 0, opacity: 0.7}
```



- Couleurs catégorielles :

- ***d3.schemeCategory10***

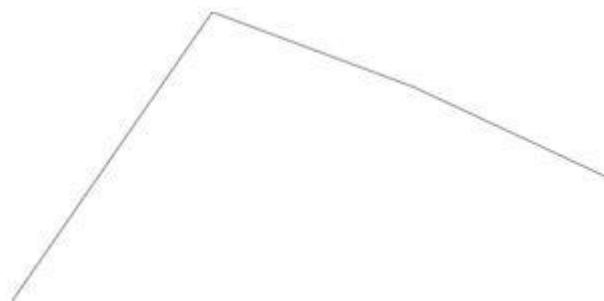


- ***d3.schemeCategory20***



Formes : lignes (1)

```
var tweets = [  
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},  
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},  
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},  
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}  
];  
  
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);  
var yScale = d3.scaleLinear().domain([0, 43]).range([200, 0]);  
  
var line = d3.line()  
  .x(function(d, i) {  
    return xScale(i);  
  })  
  .y(function(d) {  
    return yScale(d.Partage);  
  });  
  
d3.select('svg')  
  .append('path')  
  .attr("fill", "none")  
  .attr("stroke", "steelblue")  
  .attr('d', line(tweets));
```



```
<path fill="none" stroke="steelblue" d="M0,144.18604651162792L100,0L200,37.209302325581405L300,83.72093023255813"></path>
```

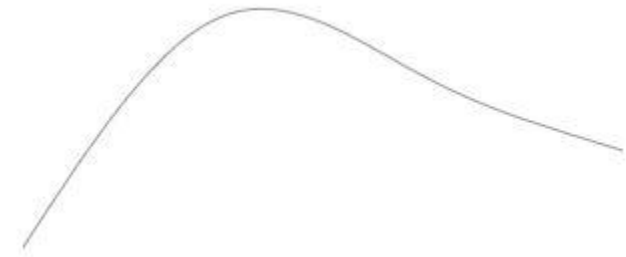

Formes : lignes (2)

```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}
];

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})
var hauteur = 200;
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);
var yScale = d3.scaleLinear().domain([0, 43]).range([hauteur - marge.bas, marge.haut]);

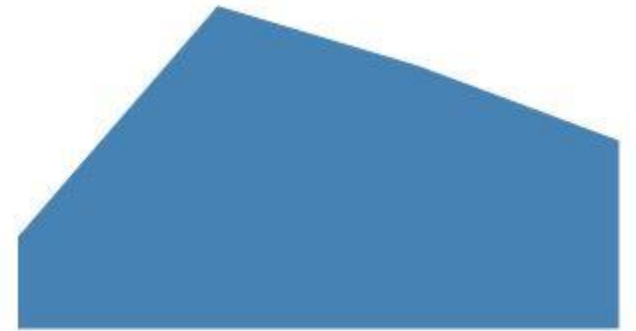
var line = d3.line()
  .curve(d3.curveNatural)
  .x(function(d, i) {
    return xScale(i);
  })
  .y(function(d) {
    return yScale(d.Partage);
  });

d3.select('svg')
  .append('path')
  .attr("fill", "none")
  .attr("stroke", "steelblue")
  .attr('d', line(tweets));
```



Formes : zones (1)

```
var tweets = [  
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},  
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},  
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},  
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}  
];  
  
var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})  
var hauteur = 200;  
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);  
var yScale = d3.scaleLinear().domain([0, 43]).range([hauteur - marge.bas, marge.haut]);  
  
var area = d3.area()  
  .x(function(d, i) {  
    return xScale(i);  
  })  
  .y0(yScale(0))  
  .y1(function(d) {  
    return yScale(d.Partage);  
  });  
  
d3.select('svg')  
  .append('path')  
  .attr("fill", "steelblue")  
  .attr("stroke", "steelblue")  
  .attr('d', area(tweets));
```



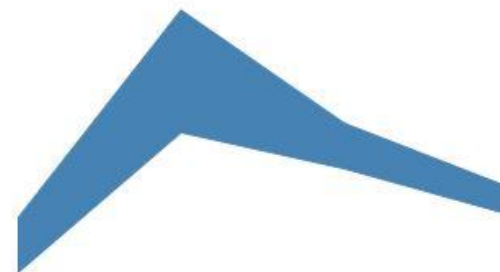
Formes : zones (2)

```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})
var hauteur = 200;
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);
var yScale = d3.scaleLinear().domain([d3.min(tweets, d => d.Partage), d3.max(tweets, d => d.Like)])
  .range([hauteur - marge.bas, marge.haut]);

var area = d3.area()
  .x(function(d, i) {
    return xScale(i);
  })
  .y0(function(d) {
    return yScale(d.Partage);
  })
  .y1(function(d) {
    return yScale(d.Like);
  });

d3.select('svg')
  .append('path')
  .attr("fill", "steelblue")
  .attr("stroke", "steelblue")
  .attr('d', area(tweets));
```



Formes : empilées (1)

```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})
var hauteur = 200;
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 45]);
var yScale = d3.scaleLinear().domain([0, 113]).range([hauteur - marge.bas, marge.haut]);

var empile = d3.stack()
  .keys(['Partage', 'Like']);

d3.select('svg')
  .selectAll('g')
  .data(empile(tweets)) // [ [0, 12], [0, 43], [0, 35], [0, 25] ], [12, 36], [43, 113], [35, 80], [25, 56] ]
  .enter()
  .append('g')
  .style('fill', (d, i) => d3.schemeCategory10[i])
  .selectAll('rect')
  .data(d => d)
  .enter()
  .append('rect')
  .attr('width', 14)
  .attr('x', (d, i) => xScale(i))
  .attr('y', d => yScale(d[1]))
  .attr('height', d => yScale(d[0]) - yScale(d[1]));
```



Formes : empilées (2)

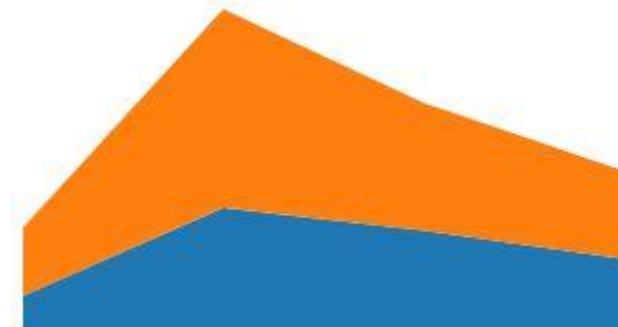
```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})
var hauteur = 200;
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);
var yScale = d3.scaleLinear().domain([0, 113]).range([hauteur - marge.bas, marge.haut]);

var area = d3.area()
  .x(function(d, i) {
    return xScale(i);
  })
  .y0(function(d) {
    return yScale(d[0]);
  })
  .y1(function(d) {
    return yScale(d[1]);
  });

var empile = d3.stack()
  .keys(['Partage', 'Like']);

d3.select('svg')
  .selectAll('path')
  .data(empile(tweets))
  .enter()
  .append('path')
  .style('fill', (d, i) => d3.schemeCategory10[i])
  .attr('d', area);
```



Formes : empilées (3)

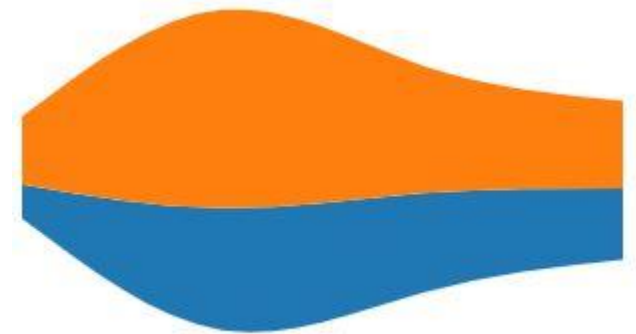
```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30});
var hauteur = 200;
var xScale = d3.scaleLinear().domain([0, 3]).range([0, 300]);
var yScale = d3.scaleLinear().domain([0, 113]).range([hauteur - marge.bas, marge.haut]);

var area = d3.area()
  .x(function(d, i) {
    return xScale(i);
  })
  .y0(function(d) {
    return yScale(d[0]);
  })
  .y1(function(d) {
    return yScale(d[1]);
  })
  .curve(d3.curveNatural);

var empile = d3.stack()
  .keys(['Partage', 'Like'])
  .offset(d3.stackOffsetWiggle);

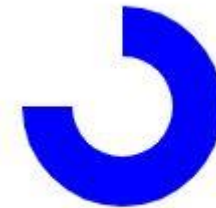
d3.select('svg')
  .selectAll('path')
  .data(empile(tweets))
  .enter()
  .append('path')
  .style('fill', (d, i) => d3.schemeCategory10[i])
  .attr('d', area);
```



Formes : portions (1)

```
var pi = Math.PI;
var arcPartage = d3.arc()
  .innerRadius(25)
  .outerRadius(50)
  .startAngle(0)
  .endAngle((0.75*360)*(pi/180)); //conversion degré ==> radian

d3.select('g')
  .append('path')
  .attr('d', arcPartage);
```



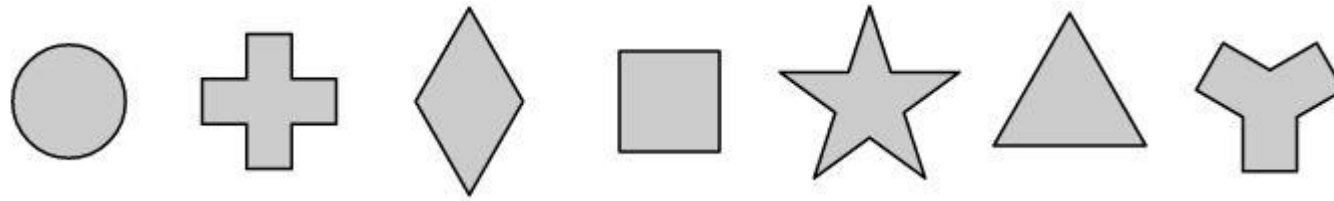
Formes : portions (2)

```
var tweets = [  
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},  
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},  
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},  
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}  
];  
  
var pi = Math.PI;  
  
var arcPartage = d3.arc()  
  .innerRadius(25)  
  .outerRadius(50)  
  .padAngle(.03);  
  
var piePartage = d3.pie().sort(null).value(function(d) {  
  return d.Partage;  
});  
  
var arcs = d3.select('svg').append("g")  
  .selectAll("arc")  
  .data(piePartage(tweets))  
  .enter()  
  .append('path')  
  .attr('d', arcPartage)  
  .style('fill', (d, i) => d3.schemeCategory10[i])  
  .attr("transform", "translate( 100, 100)");
```



Formes : symboles

- `d3.symbol()`



Axes

```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

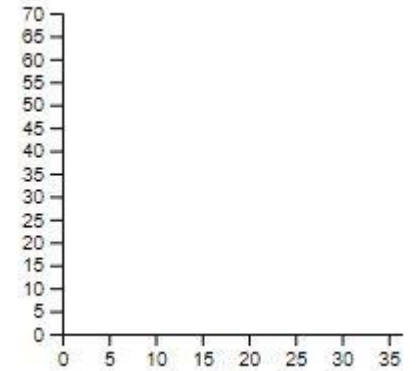
var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30})
var hauteur = 200;
var largeur = 200;
var xScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([0, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Like)]).range([hauteur - marge.bas, marge.bas]);

var x_axis = d3.axisBottom().scale(xScale);

var y_axis = d3.axisLeft().scale(yScale);

d3.select('svg').append('g')
  .attr("transform", "translate(" + marge.gauche + ", " + 0 + ")")
  .call(y_axis);

d3.select('svg').append('g')
  .attr("transform", "translate(" + marge.gauche + ", " + (hauteur - marge.bas) + ")")
  .call(x_axis);
```

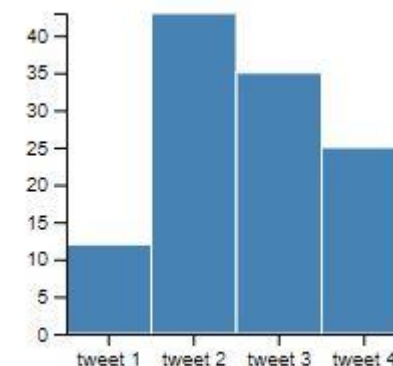


Histogramme

```
var marge = {haut: 20, droite: 30, bas: 20, gauche: 30};
var hauteur = 200;
var largeur = 200;
var xScale = d3.scaleBand().domain(tweets.map(d => d.Texte)).range([marge.gauche, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([hauteur - marge.bas, marge.haut]);
var xAxis = g => g
    .attr("transform", `translate(0,${hauteur - marge.bas})`)
    .call(d3.axisBottom(xScale).tickSizeOuter(0));
var yAxis = g => g
    .attr("transform", `translate(${marge.gauche}, 0)`)
    .call(d3.axisLeft(yScale));

d3.select('svg').selectAll("rect")
    .data(tweets)
    .enter()
    .append("rect")
    .style("fill", "steelblue")
    .style("stroke", "white")
    .attr("y", function(d) {
        return yScale(d.Partage);
    })
    .attr("height", function(d) {
        return yScale(0) - yScale(d.Partage);
    })
    .attr("width", xScale.bandwidth() - 1)
    .attr("x", function(d, i) {
        return xScale(d.Texte);
    });

d3.select('svg').append("g").call(xAxis);
d3.select('svg').append("g").call(yAxis);
```



Nœud-lien

```
var tweetsReseau = {
  noeuds: [{"Twittos": "Toto1"}, {"Twittos": "Toto2"}, {"Twittos": "Toto3"}, {"Twittos": "Toto4"}],
  liens: [{"source": "Toto1", "target": "Toto2", "Partage": 1},
          {"source": "Toto1", "target": "Toto3", "Partage": 2},
          {"source": "Toto2", "target": "Toto3", "Partage": 6},
          {"source": "Toto3", "target": "Toto4", "Partage": 3}]
};

var marge = ({haut: 20, droite: 30, bas: 20, gauche: 30}); var hauteur = 200; var largeur = 200;

var graphe = d3.forceSimulation()
  .force("charge", d3.forceManyBody().strength(-200))
  .force("link", d3.forceLink().id(function(d) { return d.Twittos; }).distance(40))
  .force("x", d3.forceX(largeur / 2)).force("y", d3.forceY(hauteur / 2))
  .on("tick", ticked);

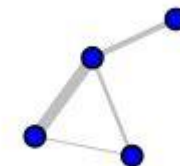
var lien = d3.select('svg').selectAll(".link"); var noeud = d3.select('svg').selectAll(".node");

graphe.nodes(tweetsReseau.noeuds);
graphe.force("link").links(tweetsReseau.liens);

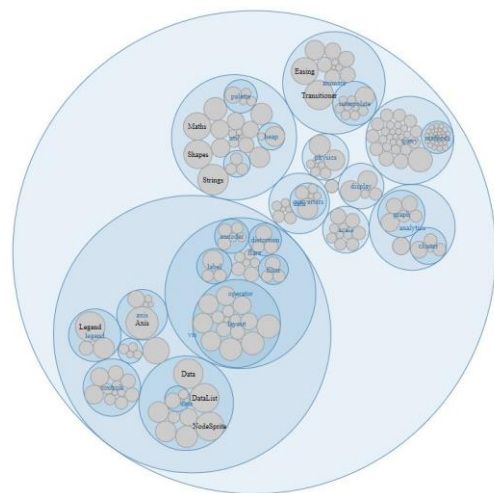
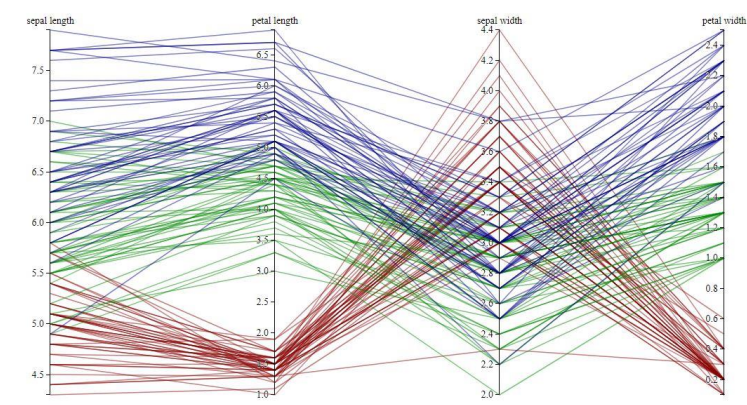
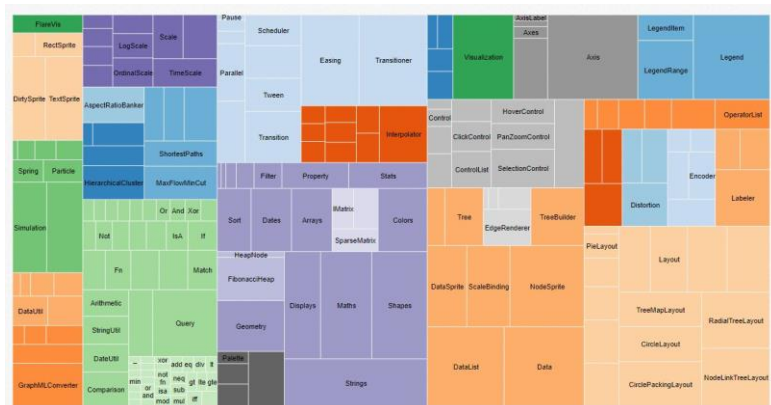
lien = lien.data(tweetsReseau.liens)
  .enter().append("line")
  .style("stroke", "grey").style("stroke-opacity", 0.5)
  .attr("class", "link").style("stroke-width", d => d.Partage);

noeud = noeud.data(tweetsReseau.noeuds)
  .enter().append("circle")
  .attr("class", "node")
  .attr("r", 5).style("stroke", "black").style("stroke-width", 1.5).style("fill", 'blue');

function ticked() {
  lien.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; });
  noeud.attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; });
}
```



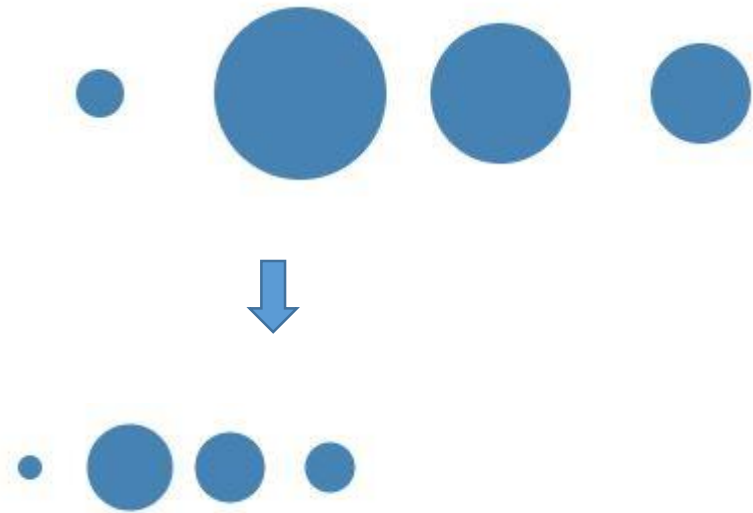
Et encore ...



Partie 2

Transitions (1)

```
var tweets = [  
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12"},  
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43"},  
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35"},  
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25"}  
];  
  
var s = d3.select('g')  
  .selectAll('circle')  
  .data(tweets)  
  .enter().append('circle')  
  .attr('transform', function(d, i) {  
    return "translate("+ (i+1) * 100 +", 100)";  
  })  
  .attr('r', function(d) { return d.Partage; });  
  
setTimeout(function() {  
  s.transition()  
    .duration(1000)  
    .attr('transform', function(d, i) {  
      return "translate("+ (i+1) * 50 +", 100)";  
    })  
    .attr("r", function(d) {  
      return d.Partage/2;  
    });  
}, 3000);
```



Transitions (2)

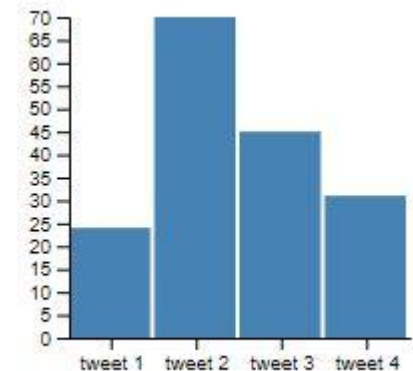
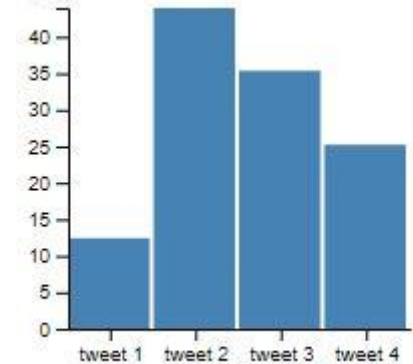
```
var tweets = [
  {"Texte": "tweet 1", "Twittos": "Toto 1", "Partage": "12", "Like": "24"},
  {"Texte": "tweet 2", "Twittos": "Toto 2", "Partage": "43", "Like": "70"},
  {"Texte": "tweet 3", "Twittos": "Toto 3", "Partage": "35", "Like": "45"},
  {"Texte": "tweet 4", "Twittos": "Toto 4", "Partage": "25", "Like": "31"}
];

var marge = {haut: 20, droite: 30, bas: 20, gauche: 30}; var hauteur = 200; var largeur = 200;
var xScale = d3.scaleBand().domain(tweets.map(d => d.Texte)).range([marge.gauche, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([hauteur - marge.bas, marge.haut]);
var xAxis = g => g.attr("transform", `translate(0,${hauteur - marge.bas})`)
    .call(d3.axisBottom(xScale).tickSizeOuter(0));
var yAxis = g => g.attr("transform", `translate(${marge.gauche}, 0)`)
    .call(d3.axisLeft(yScale));

var bars = d3.select('svg').selectAll("rect")
    .data(tweets)
    .enter()
    .append("rect")
    .style("fill", "steelblue")
    .style("stroke", "white")
    .attr("y", d => yScale(d.Partage))
    .attr("height", d => yScale(0) - yScale(d.Partage))
    .attr("width", xScale.bandwidth() - 2)
    .attr("x", d => xScale(d.Texte));

var abscisses = d3.select('svg').append("g").call(xAxis);
var ordonnees = d3.select('svg').append("g").call(yAxis);

setTimeout(function() {
  yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Like)]).range([hauteur - marge.bas, marge.haut]);
  yAxis = g => g.attr("transform", `translate(${marge.gauche}, 0)`)
    .call(d3.axisLeft(yScale));
  ordonnees.transition().duration(1000).delay(500).call(yAxis);
  bars.transition().duration(1000).delay(500)
    .attr("y", d => yScale(d.Like))
    .attr("height", d => yScale(0) - yScale(d.Like));
}, 3000);
```



Interactions (1) : cliquer

- Cliques :

- Bouton gauche de la souris

```
.on('click', function(d) {  
    // gestion de l'évènement clique gauche  
})
```

- Bouton droit de la souris

```
.on("contextmenu", function(d) {  
    d3.event.preventDefault();  
    // gestion de l'évènement clique droit  
    d3.event.stopPropagation();  
})
```

- Touche du clavier : P. ex., la touche « alt » et « ctrl »

```
if (d3.event.altKey) {  
    // gestion de l'évènement touche "alt"  
} else if (d3.event.ctrlKey) {  
    // gestion de l'évènement touche "ctrl"  
}
```

Interactions (2) : zoomer

- Double-clic gauche de la souris ou faire défiler avec la molette de la souris

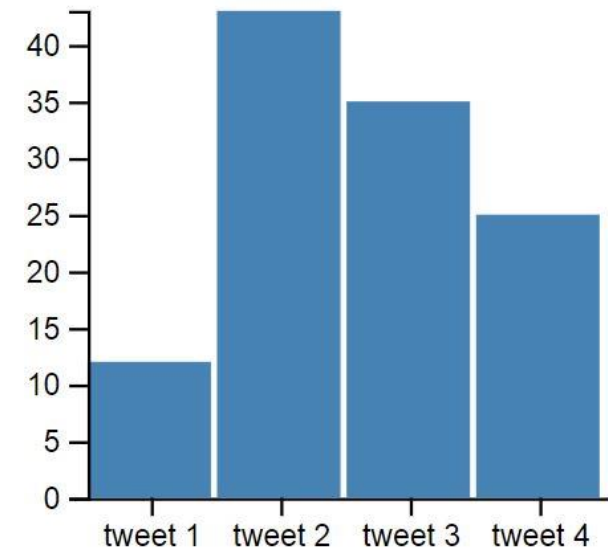
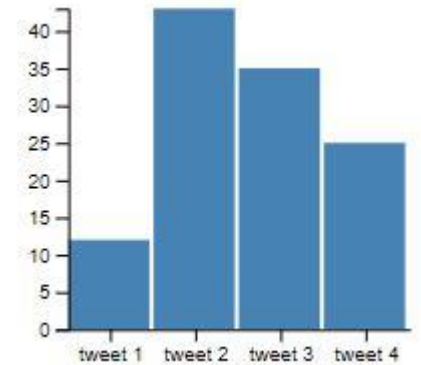
```
var marge = {haut: 20, droite: 30, bas: 20, gauche: 30}; var hauteur = 200; var largeur = 200;
var xScale = d3.scaleBand().domain(tweets.map(d => d.Texte)).range([marge.gauche, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([hauteur - marge.bas, marge.haut]);
var xAxis = g => g.attr("transform", `translate(0,${hauteur - marge.bas})`)
    .call(d3.axisBottom(xScale).tickSizeOuter(0));
var yAxis = g => g.attr("transform", `translate(${marge.gauche},0)`)
    .call(d3.axisLeft(yScale));

var zoom = d3.zoom()
    .scaleExtent([1, 100])
    .on("zoom", zoomer);

var svg = d3.select('svg').call(zoom);
var bars = svg.selectAll("rect")
    .data(tweets)
    .enter()
    .append("rect")
    .style("fill", "steelblue")
    .style("stroke", "white")
    .attr("y", d => yScale(d.Partage))
    .attr("height", d => yScale(0) - yScale(d.Partage))
    .attr("width", xScale.bandwidth() - 2)
    .attr("x", d => xScale(d.Texte));

var abscisses = svg.append("g").call(xAxis);
var ordonnees = svg.append("g").call(yAxis);

function zoomer() {
    svg.attr("transform", d3.event.transform);
}
```



Interactions (3) : glisser-déposer

```
var marge = {haut: 20, droite: 30, bas: 20, gauche: 30}; var hauteur = 200; var largeur = 200;
var xScale = d3.scaleBand().domain(tweets.map(d => d.Texte)).range([marge.gauche, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([hauteur - marge.bas, marge.haut]);
var xAxis = g => g.attr("transform", `translate(0, ${hauteur - marge.bas})`)
    .call(d3.axisBottom(xScale).tickSizeOuter(0));
var yAxis = g => g.attr("transform", `translate(${marge.gauche}, 0)`)
    .call(d3.axisLeft(yScale));

var glisse = d3.drag()
    .subject(function (d) { return d; })
    .on("start", attraper)
    .on("drag", glisser)
    .on("end", déposer);

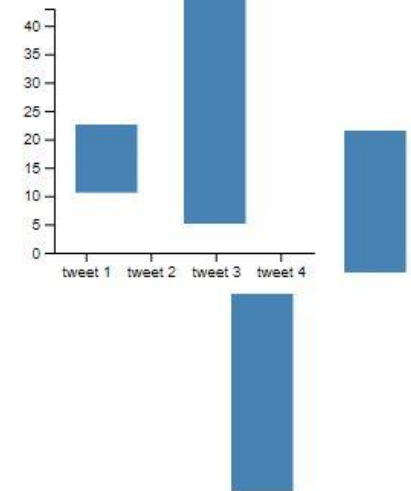
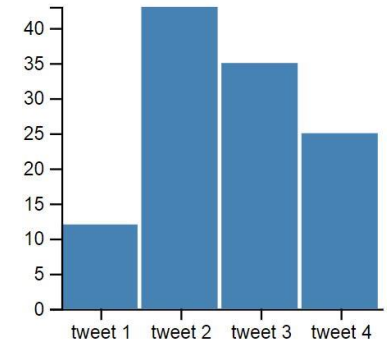
var svg = d3.select('svg');
var bars = svg.selectAll("rect")
    .data(tweets)
    .enter()
    .append("rect")
    .style("fill", "steelblue")
    .style("stroke", "white")
    .attr("y", d => yScale(d.Partage))
    .attr("height", d => yScale(0) - yScale(d.Partage))
    .attr("width", xScale.bandwidth() - 2)
    .attr("x", d => xScale(d.Texte))
    .call(glisse);

var abscisses = svg.append("g").call(xAxis); var ordonnees = svg.append("g").call(yAxis);

function attraper(d) {
    d3.event.sourceEvent.stopPropagation();
    d3.select(this).classed("dragging", true);
}

function glisser(d) {
    d3.select(this).attr("x", d.x = d3.event.x).attr("y", d.y = d3.event.y);
}

function déposer(d) {
    d3.select(this).classed("dragging", false);
}
```



Interactions (4) : zone de sélection

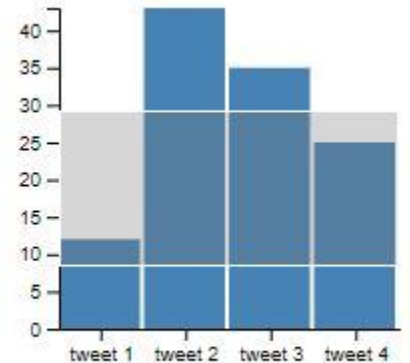
```
var marge = {haut: 20, droite: 30, bas: 20, gauche: 30}; var hauteur = 200; var largeur = 200;
var xScale = d3.scaleBand().domain(tweets.map(d => d.Texte)).range([marge.gauche, largeur]);
var yScale = d3.scaleLinear().domain([0, d3.max(tweets, d => d.Partage)]).range([hauteur - marge.bas, marge.haut]);
var xAxis = g => g.attr("transform", `translate(0,${hauteur - marge.bas})`)
    .call(d3.axisBottom(xScale).tickSizeOuter(0));
var yAxis = g => g.attr("transform", `translate(${marge.gauche},0)`)
    .call(d3.axisLeft(yScale));

var zone = d3.brush().extent([[0, 0], [hauteur, largeur]]);
var svg = d3.select('svg');

var bars = svg.selectAll("rect")
    .data(tweets)
    .enter()
    .append("rect")
    .style("fill", "steelblue")
    .style("stroke", "white")
    .attr("y", d => yScale(d.Partage))
    .attr("height", d => yScale(0) - yScale(d.Partage))
    .attr("width", xScale.bandwidth() - 2)
    .attr("x", d => xScale(d.Texte));

var abscisses = svg.append("g").call(xAxis);
var ordonnees = svg.append("g").call(yAxis);

svg.call(zone)
```



Application : Info-RSN



Détection et visualisation de communautés de Twitter



Application : Info-RSN

Contexte

- 1 million d'articles générant 17 millions de tweets de 1,5 million de tweetos
- Trois relations sociales étudiées : le partage, la mention et le suivi



Objectifs

- Identifier et visualiser la structure des communautés afin de comprendre la circulation de l'information dans les RSN
- Déterminer le rôle des membres des communautés en fonction du profil : actif, média ou robot



Application : Info-RSN

Communautés statiques et mono-relation

- Modélisation sous forme de graphes : les nœuds représentent les tweetos tandis que les liens représentent une relation sociale
- Algorithme multi-niveaux ayant une triangulation maximale (problème NP-difficile et APX-difficile) comme structure de départ
- Visualisations interactives répondant à des tâches d'analyses spécifiques via l'outil *NLCOMS (Nœud-Lien et COMMunautésS)*

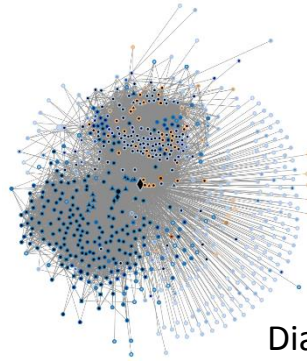
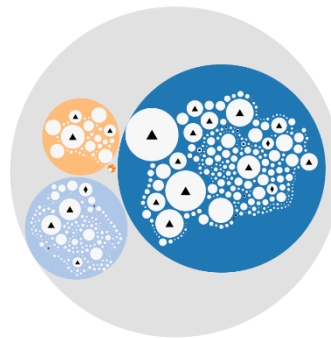
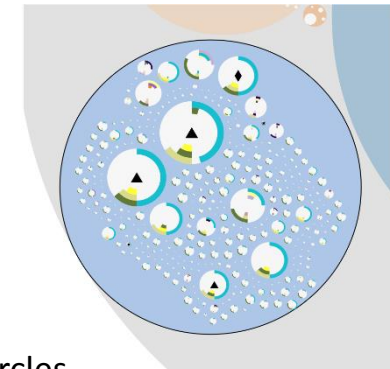


Diagramme nœud-lien



Emboîtement de cercles

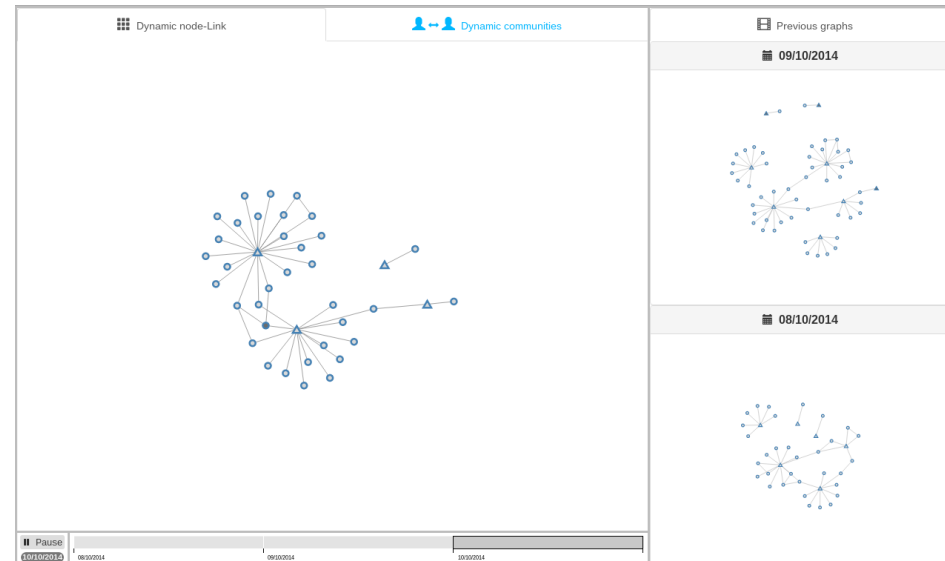


Application : Info-RSN

Communautés dynamiques

- **Graphes dynamiques**

Détection dynamique
VS
Détection statique



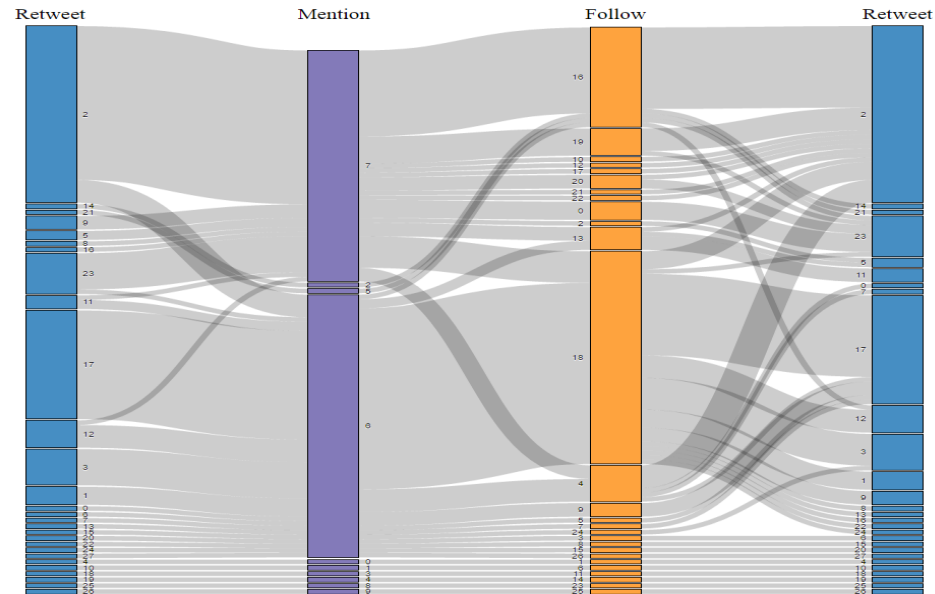
- Ré-identification locale des communautés au fil du temps
- Utilisation des animations et du positionnement ancré

Application : Info-RSN

Communautés multi-relationnelles

- **Multi-graphes**

*Détection multi-relationnelles
VS
Détection mono-relationnelle*



- Détection de communautés multi-relationnelles paramétrable
- Analyser le chevauchement entre les différentes structures

Références

- <https://d3js.org/>
- <https://github.com/d3/d3/wiki/Tutorials>
- <https://bost.ocks.org/mike/>
- <https://d3indepth.com/>
- Ændrew Rininsland and Swizec Teller, **D3.js 4.x Data Visualization - Third Edition**, Packt, 2017
- Jos Dirksen, **Expert Data Visualization**, Packt, 2017