Visualisation de données avec d3.js – TP2

Arthur Katossky

07/03/2019

SVG (~ 45 min)

SVG est un langage de description d'images vectorielles, c.-à-d. décrites par des courbes paramétrées, et non décomposées en pixels. Il est particulièrement bien adapté à la description de graphiques à partir de formes simples, comme des lignes, des cercles, des rectangles, etc. pouvant changer d'échelle sans perdre en qualité, d'où le nom du format : *scalable vector graphics*.

SVG peut s'intégrer directement dans une page HTML, à l'aide de la balise double `<svg> ... </svg>`. Dans ce codePen (https://codepen.io/katossky/pen/QoGyWP), l'image initiale a été remplacée par l'image SVG suivante :

```
<svg id="graphic">
  <circle cx="50%" cy="90%" r="20" fill="blue"/>
  <circle cx="100" cy= "40" r="10" stroke="orange" stroke-width="4"/>
  <circle cx="150" cy="100" r="40" stroke="purple" stroke-width="10" />
  <circle cx="110" cy="200" r="90" opacity="0.5" />
  <circle cx= "50" cy= "80" r="20"/>
  <circle cx="350" cy="300" r="30"/>
  <g>
    <line x1="0" y1="0" x2="1000" y2="1000" style="stroke:red;stroke-width:2" />
    <text x="20" y="-5" fill="red" transform="rotate(45)">Une tendance claire</text>
  </g>
</svg>
```

1. À quoi correspondent les paramètres `cx`, `cy`, `r`, `fill`, `stroke`, `opacity` et `stroke-width` de l'objet `circle` ?
2. Est-il possible d'utiliser CSS pour changer la couleur des disques à `indigo` ? Laquelle des deux sources de style prend le pas sur l'autre ? (La réponse vaut pour les autres paramètres.)
3. Écrivez une fonction javascript `toggle_circle` qui ajoute/enlève (EN : *toggle*) la classe `selected` aux disques à chaque fois que l'utilisateur clique dessus. Créez un style ditinctif pour la classe `selected`. Abonnez cette fonction à l'événement `click` sur le premier `circle` de l'objet `svg`. (20 min)
4. En utilisant du code CSS, faîtes apparaître le texte uniquement au survol de la ligne. (***Indice*** **:** utilisez la propriété `opacity` et le sélecteur CSS `+`.) (15 min)
5. Quel est le rôle de la balise `<g>...</g>` ?

Pour patienter : a. Faîtes en sorte que l'apparition/disparition de texte soit moins abrupte. b. Trouvez une solution pour que l'apparition du texte se déclenche à proximité de la ligne, pas uniquement quand la souris est exactement dessus. (***Indice*** **:** pensez transparence… et n'hésitez pas à modifier le SVG !)

Proposition de correction : https://codepen.io/katossky/pen/LaxVGr

Automatisation de la construction (~ 30 min)

En réalité, nous voulons utiliser le SVG comme canal de représentation de données. Dans le codePen (https://codepen.io/katossky/pen/RdKraj) [1], les données sont fournies "en dur" dans le code JavaScript : une liste `countries` d'objets avec comme attributs : `x`, `y`, `text` et `color`.

1. Le CSS a aussi été changé, ce qui explique la "disparition" de certains éléments de l'image SVG.

Par exemple, `countries[0]` vaut :

```
{
  x: 102,
  y: 345,
  text: "Pays 1"
  color: "#54BC9A"
}
```

Cette structure de données utilisée en Javascript est appelée `JSON`. La propriété `x` de l'objet `countries[0]` est à son tour accessible par `countries[0].x` ou `countries[0]["x"]` [2].

6. Supprimez la balise `svg` du code HTML et son contenu ; créez une balise `svg` vide au même endroit à l'aide du code Javascript. (5 min)
   Remarque : contrairement à des objets HTML classiques, crées avec `document.createElement(`*`type`*`)`, les objets SVG doivent être crées avec `document.createElementNS("http://www.w3.org/2000/svg",`*`type`*`)`.)
7. À partir de l'objet `countries`, générez autant d'objets SVG (balise `<g>...</g>`) contenant chacun un cercle de rayon 5 et un texte associé. La position est celle des données. La couleur affecte la bordure du cercle et le texte. Le texte doit être lisible. (15 min)
8. Abonnez la fonction `toggle_circle` à l'événement `click` sur chacun des cercles. (10 min)
   Remarque : la syntaxe `for(`*`element of selection`*`)` permet de parcourir une sélection élément par élément.)

Pour patienter : c. Générez des nouvelles positions pour `x` et `y` à chaque fois que l'utilisateur clique sur "graphique aléatoire".

Proposition de correction : https://codepen.io/katossky/pen/jJyqVq (sans la question c)

Syntaxe simplifiée avec d3.js (~ 30 min)

Dans ce nouveau codePen (https://codepen.io/katossky/pen/aMpJmN), un lien vers la bibliothèque `d3.js` a été ajouté, et la syntaxe JavaScript classique a été partiellement remplacée par la syntaxe typique de `d3.js`, plus concise. (L'ancien est code est présent en commentaire sur le codePen.)

```
let svg = d3.select("#content")
  // convention: 1 indentation => pas le même objet du DOM
  .append("svg");

for(c of countries){svg.append("g")};
// ↑ Cette syntaxe n'est pas très "d3.js"! Nous l'améliorerons plus tard.
let groups  = svg.selectAll("g");

let circles = groups
  .append("circle")
     // convention: 2 indentations => même objet du DOM
    .attr("r", 10)
    .attr("cx",     (d,i) => countries[i].x) // ignorez le "d" pour l'instant
    .attr("cy",     (d,i) => countries[i].y)
    .attr("stroke", (d,i) => countries[i].color);

circles.on("click", toggle_circle);
```

2. Un ojbet JSON peut tout à fait contenir un emboîtement infini de listes (ensemble d'éléments ordonnés, non nommés, indexés à partir de 0, entre crochets) et d'objets (ensemble d'éléments non ordonnés, indexés par des noms appelés propriétés, attributs ou clés, listés entre accolades).

9. Pour sélectionner un ou plusieurs éléments du DOM, `d3.js` possède les syntaxes `d3.select(`*`selector`*`)` et `d3.selectAll(`*`selector`*`)`, où *`selector`* est un sélecteur CSS. Cette syntaxe remplace les méthodes classiques `document.getElementById`, `document.getElementsByTag` et d'autres encore. Trouvez les 2 utilisations dans le code ci-dessus. Quelle est la différence entre les deux ? Que signifie la syntaxe *`object`*`.selectAll(`*`selector`*`)`, comme dans `svg.selectAll("g")` ?
10. Pour ajouter un élément au DOM, `d3.js` possède la syntaxe *`object`*`.append(`*`type`*`)` (créer un nouvel objet de type *`type`* comme dernier enfant de l'objet *`object`*). Expliquer ce que produit l'instruction `let svg = d3.select("#content").append("svg");`.
11. `append` fonctionne également avec une sélection : la méthode ajoute un nouvel objet de type *`type`* comme dernier enfant de *chaque* objet de la sélection. Que fait le code : `let circles = groups.append("circle");` ?
12. L'avantage des syntaxes `d3.js` est qu'elles ne se réduisent pas à un simple tableau d'éléments du DOM :
    — elles permettent d'agir sur *tous* les éléments de la sélection à la fois, soit avec la même action, soit avec une action distincte pour chaque objet
    — elles renvoient toujours une sélection, soit la même (`text`, `attr`, etc.), soit une autre (`select`, `append`, etc.), ce qui permet de *chaîner* les méthodes
    Trouvez un exemple de chaque situation dans le code.
13. Réimplémentez les étiquettes des points en utilisant cette nouvelle syntaxe. (15 min)

Proposition de correction : https://codepen.io/katossky/pen/ZPLyOR

Intégration des données avec `d3.js` (~ 15 min)

`d3.js` possède également une syntaxe pour faciliter la liaison entre les objets du DOM et les données. Par exemple, dans la section précédente, si on implémente une fonction qui retire des pays du graphique en retirant des `<circle>`s du DOM, on perdrait la correspondance entre les listes `circles` (qui ne contient pas les objets retirés) et `countries` (qui n'est pas affecté). Une possibilité serait de mettre à jour l'objet `countries`, mais `d3.js` nous offre plus de flexibilité encore en stockant les données au niveau du DOM lui-même ! **Nouveau code :** https://codepen.io/katossky/pen/MxJovB

```
let svg = d3.select("#")
  .append("svg");

let groups  = svg
  .selectAll("g")    // 1. Il n'existe aucune balise <g> à ce stade.
    .data(countries) // 2. L'objet "countries" est lié à la sélection.
    .join("g")       // 3. Une balise <g> est ajoutée par objet dans `countries`.

    .attr("fill",   d => d.color); // 4. Ensuite la vie continue...

// ... sauf qu'à partir de maintenant, tous les <g> et tous les objets enfants de chaque <g>
// ont accès à l'objet correspondant de la liste "countries" (appelé le "datum"
// de <g>). Peu importe si entre temps l'ordre à changé, ou si certains <g> ont
// été supprimés. Dans la fonction "d => d.color" fait référence au "datum".

let circles = groups
  .append("circle")
    .attr("cx",     d => d.x)
    .attr("cy",     d => d.y)
    .attr("stroke", d => d.color);
```

```
groups
  .append("text")
    .attr("x",      d => d.x+10)
    .attr("y",      d => d.y+5)
    .text(d => d.text);
```

14. Réordonnez la sélection `circles` avec la méthode `sort`, entre le moment où les cercles sont placés dans le plan et le moment où leur couleur leur est donnée. (`sort` exige une fonction `function(a,b)` qui retourne un nombre positif si l'élément `a` est "plus grand" que `b` et un nombre négatif sinon.) Que constatez-vous ? (10 min)

Proposition de correction : https://codepen.io/katossky/pen/jJywRv

Échelles, couleurs et axes (~ 30 min)

En réalité, les données ne vous sont pas communiquées à une échelle directement adaptée à la représentation (distances en pixels, couleurs, etc.). Autrement dit, le `i`-ème objet de la liste `countries` ressemble plus souvent à ceci :

```
{
  name: "Yemen"
  life_expectancy: 63.27,
  income_per_person: 2380,
  population: 28036829,
  continent: "Asia"
}
```

… qu'aux données avec lesquelles nous travaillions jusqu'à présent. **Ce nouveau codePen** (https://codepen.io/katossky/pen/ywgzbV) donne l'objet `countries` dans un format plus classique.

`d3.js` fournit une interface utile pour calculer le positionnment des figurés selon les différents canaux de visualisation choisis (position, longueur, etc.). L'idée est d'à chaque fois définir une fonction de conversion, `channel(value)`, qui permette de calculer les valeurs du canal (par ex. les ordonnées `y`) associé à chaque donnée (par ex. l'espérance de vie) :

```
var y = d3.scaleLinear() // Correspondance linéaire entre:
  .domain([40,100])      // - domaine de validité des données: 40-100 ans
  .range(0, 800)         // - étendue de l'axe des ordonnées: 0-800 px
```

Notre code devient donc :

```
let content = d3.select("#content"),
    svg     = content.append("svg");

let height  = content.node().offsetHeight, // hauteur de la zone d'affichage
    width   = content.node().offsetWidth,  // largeur de la zone d'affichage
    inner_margin = 30, // marge entre les axes et le premier point
    outer_margin = 30, // marge entre le graphique lui-même et le bord
    margin = inner_margin+outer_margin;

let myColors = ["#6EBB87", "#2CB8EA", "#DA94CE", "#DE9D6C"];
// couleurs de: http://bl.ocks.org/katossky/af9e0f577c5a5aaa9c254cf2dd4b5b96

var x = d3.scaleLinear()
      .domain(..............)
```

```
      .range(..............),
    y = d3.scaleLinear()
      .domain([50,85]) // étendue établie par tatonnement
      .range([margin, width-margin]),
    o = d3.scaleOrdinal()
      .domain(.................)
      .range(..................);

let circles = groups
    // .sort((a,b) => b.population-a.population)
  .append("circle")
    .attr("cx",     ........................)
    .attr("cy",     d => y(d.life_expectancy))
    .attr("stroke", ........................);

groups
  .append("text")
    .attr("x",      d => ......................)
    .attr("y",      d => y(d.life_expectancy)+5)
    .text(d => d.name);
```

15. Complétez le code pour l'axe des abcisses (variables continue) et pour les couleurs (variable discrète).
16. Complétez le code pour que la surface des cercles soit proportionelle à la population. ***Indice*** **:** Le rayon doit-il être proportionnel à la population ? N'hésitez pas à consulter la documentation pour trouver d'autres types d'échelles ! (https://github.com/d3/d3-scale)
17. Arrivez-vous à cliquer sur les pays placés au même niveau que la Chine ? Comprenez-vous pourquoi ? Décommentez la ligne qui commence par `.sort...` et retentez la même opération. Qu'est-ce qui a changé ?
18. L'axe des `x` peut être ajouté de cette façon :

```
let xAxis = svg
  .append("g")
    .attr("transform", "translate(0," + (height-outer_margin) + ")")
    .call(d3.axisBottom().scale(x));
```

```
Procédez de même pour l'axe de y. **Remarque:** L'attribut `transform` d'un objet SVG permet d'applique
```

En attendant :

d. Faîtes en sorte que les échelles continues (`x()`, `y()` et `r()`) s'adaptent aux données présentes à l'initialisation.
e. L'étiquette de l'axe des x peut être ajouté ainsi :

```
xAxis.append("text")
    .attr("text-anchor", "end")
    .attr("fill", "grey")
    .attr("x", -margin)
    .attr("y", 10)
    .attr("transform", "rotate(-90)")
    .text("Espérance de vie (années)");
```

```
Comment ajouter l'étiquette de l'axe des y?
```

f. Le graphique s'adapte à toute taille d'affichage. Mais s'adapte-t-il *dynamiquement* à la taille de la fenêtre lorsque celle-ci change ? Comment pourrait-on y remédier ?

Proposition de correction : https://codepen.io/katossky/pen/RdKxJM (sans question f)

Lecture des données asynchrone (~ 30 min)

Pour finir, `d3.js` possède des utilitaires pour lire des données sous différents formats (https://github.com/d3/d3-fetch). Cependant, importer des données peut prendre du temps, particulièrement si celles-ci sont situées sur un serveur distinct. Toutes les fonctions qui dépendent des données sont donc reléguées dans un `callback`, une fonction qui ne sera convoquée qu'à l'issue du chargement. Ce fonctionnement est dit **asynchrone**, dans la mesure où le reste du code peut continuer à être exécuté dans l'intervalle.

Remarque : `d3.js` n'est pas une bibliothèque de traitement de données. Je recommande d'effectuer le maximum de traitements dans un language plus adapté, comme R, et d'exporter les données dans un format classique, comme JSON ou CSV.

Dans ce nouveau codePen (https://codepen.io/katossky/pen/BbpMRX), les données sont récupérées de façon asynchrone depuis Github.

19. Placez à l'intérieur du `callback` toutes les parties du code qui font appel aux données. Comment serait-il possible de néanmoins préserver l'organisation générale du code ?

Correction : https://codepen.io/katossky/pen/QodYOy

Travail optionnel pour la prochaine fois : un commentaire interactif

Implémentez la fonctionnalité suivante : au survol du cercle représentant un pays à la souris, une aide de lecture s'affiche en bas de la zone graphique. Ex : "L'Iran (Asie) a une espérance de vie de … pour un revenu par habitan de … $ et une population de … ." Vous pourrez améliorer la mise en forme du texte, souligner la lecture graphique à l'aide de lignes pointillées, etc.