Sémiologie graphique

TP D3 (2)

Ce TP va dans un premier temps vous permettre de réviser les notions vues lors du précédent TP via l'étude d'un vrai jeu de données. Vous verrez ensuite comment sélectionner des éléments en D3.

Les données que vous allez manipuler portent sur la COVID-19. Par rapport au jeu de données initial[1], elles ont été simplifiées de façon à ne garder que les dernières valeurs de chaque pays au moment de la rédaction du TP (19/07/2020).

Exercice 1

Commencez par récupérer les fichiers exo1.html et exo1.json disponibles sur Moodle. Ouvrez le fichier JSON. Vous pouvez remarquer qu'il contient une liste de pays pour lesquels sont spécifiées différentes valeurs liées à la COVID-19. L'objectif de cet exercice est de construire un nuage de points dans lequel les points représentent les pays et ils sont placés en fonction du nombre de cas de COVID-19 (axe X) et du nombre de morts (axe Y).

En vous inspirant du TP de la séance précédente, créez les échelles et les axes du nuage de points (attribut `total_cases` pour l'axe X et attribut `total_deaths` pour l'axe Y). Les deux axes doivent être de type `scaleLinear` puisque les valeurs sont continues. De plus, vous devez définir une fonction permettant de retourner la valeur maximale pour chacune des variables afin de déterminer les intervalles du `domain` des échelles. La figure 1 montre le résultat que vous devez obtenir.

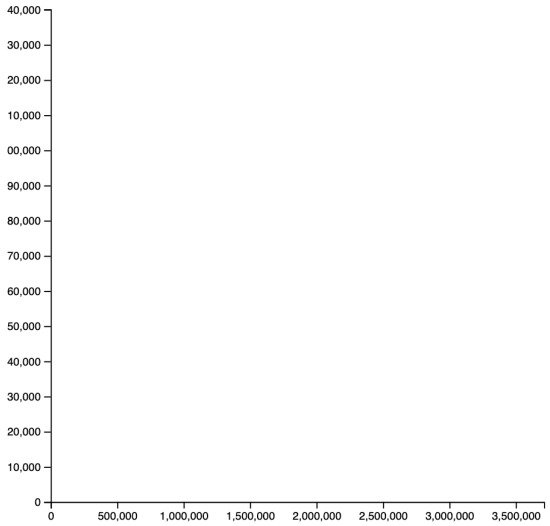

Figure 1

[1] https://github.com/owid/covid-19-data/tree/master/public/data (*Our World in Data COVID-19*)

Vous devez maintenant ajouter un petit triangle pour chaque pays. En D3, il n'existe pas d'objet permettant de créer un `polygon` SVG. Il faut donc utiliser un chemin (`path`) fermé pour chaque pays. Le code se présente sous cette forme :

```
var gTriangles = svg.append("g");
gTriangles.attr("transform","translate(40,10)");
for (var i=0; i<data.length; i++){
      var tr = gTriangles.append("path");
      tr.attr("d", triangle);
      tr.attr("fill", "teal");
      tr.attr("transform",
              "translate(" + scaleX(data[i]["total_cases"])+","
                           + scaleY(data[i]["total_deaths"])
                           + ")"
            );
}
```

Ce code ajoute un `g` nommé `gTriangles` au SVG et le positionne correctement (lignes 1 et 2). Ensuite, pour chaque pays (boucle `for`), il ajoute un chemin. L'attribut `d` contient la liste des points formant le chemin (rappelez-vous le premier TP sur SVG). Ici, on lui a passé une variable nommée `triangle` et qu'il va falloir définir. A la fin, une translation permet de positionner les triangles.

Définissez la variable `triangle` de la façon suivante avant la boucle :

```
var triangle = d3.line()([[0, 0], [3, -6], [6, 0]]);
```

La figure 2.1 montre le résultat que vous devez obtenir.

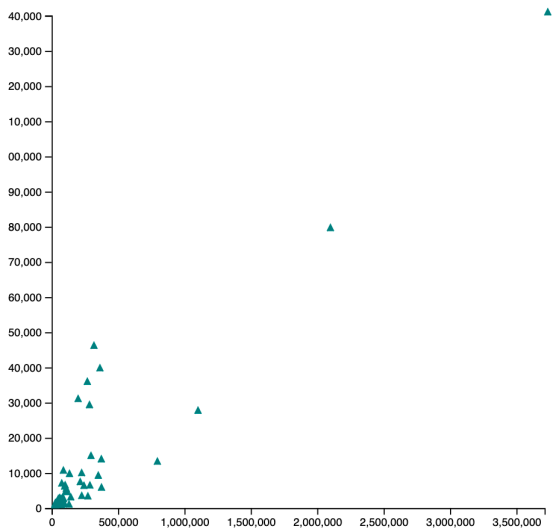

Figure 2.1

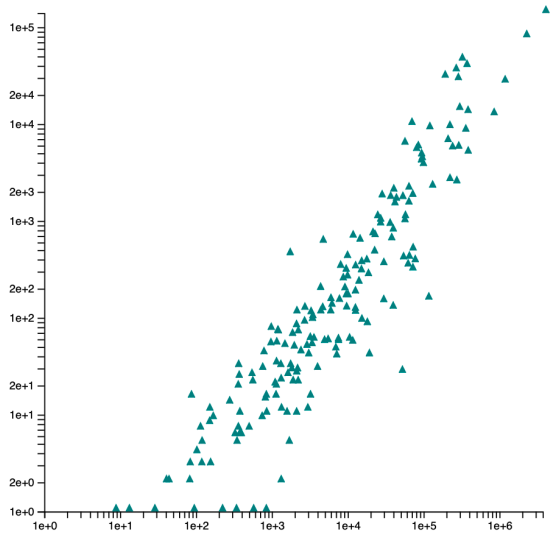

Figure 2.2

Comme vous pouvez l'observer, cette visualisation est peu lisible car la plupart des pays ont des valeurs très faibles en comparaison de certains autres. Nous allons donc devoir transformer les échelles en échelles logarithmiques. Pour cela, remplacez `d3.scaleLinear()` par `d3.scaleLog()`. Se pose alors un nouveau problème, `log(0)` n'est pas défini. Vous devez donc faire débuter les `domain` à 1 et ajouter une instruction conditionnelle dans le `for` pour que les triangles ne soient créés que si le nombre de cas et le nombre de morts sont strictement supérieurs à 0 :

```
if(data[i]["total_cases"]>0 && data[i]["total_deaths"]>0){
        ...
}
```

Vous devez obtenir l'image de la figure 2.2.

Ajoutez aussi une petite infobulle permettant, lorsque l'on survole un sommet, d'afficher la propriété `location` du fichier JSON, elle contient le nom du pays (voir figure 3.1) :

```
tr.append("title").text(data[i]["location"]);
```

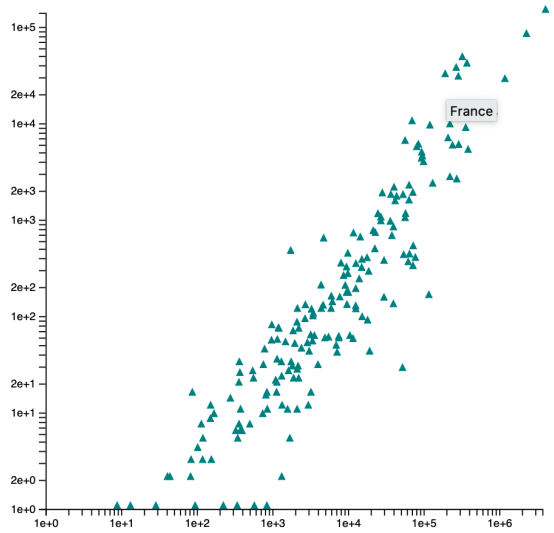

Figure 3.1

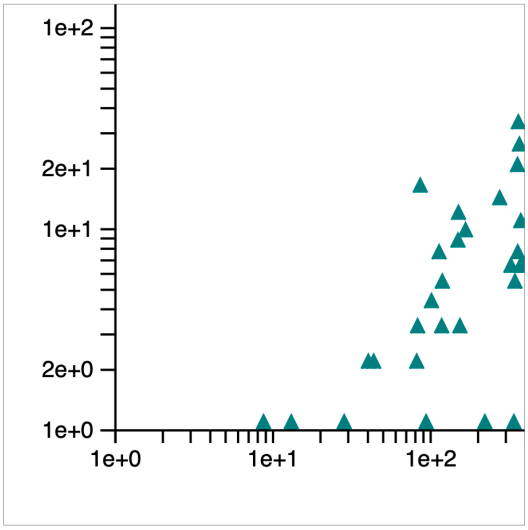

Figure 3.2

Ajoutez aussi une bordure à votre SVG ainsi qu'un zoom de façon à pouvoir explorer plus facilement votre graphique (cf. exercice 2 du TP précédent, voir figure 3.2).

Actuellement, tous les pays sont représentés à l'aide d'un triangle tourné vers le haut. Vous allez maintenant modifier votre graphique de façon à ce que les pays ayant des nouveaux cas de COVID-19 soient représentés par un triangle tourné vers le haut et les autres par un triangle tourné vers le bas. Vous devez tout d'abord créer les deux lignes correspondantes à la place de l'ancienne :

```
var triangleUp = d3.line()([[0, 0], [3, -6], [6, 0]]);
var triangleDown = d3.line()([[0, -6], [3, 0], [6, -6]]);
```

Vous devez ensuite ajouter une instruction conditionnelle lors de l'ajout de la propriété `d` au triangle, de façon à sélectionner la bonne ligne :

```
if(data[i]["new_cases"]==0){
        tr.attr("d", triangleDown);
}
else {
        tr.attr("d", triangleUp);
}
```

Vous obtenez ainsi l'image de la figure 4.1.

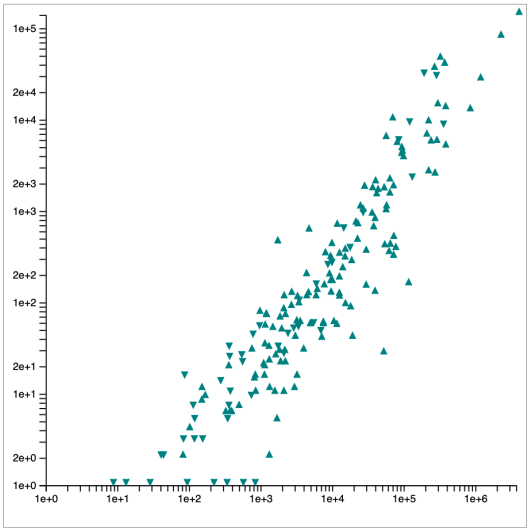

Figure 4.1

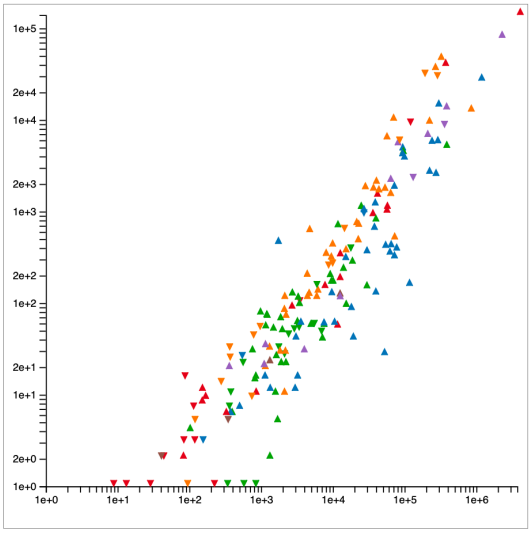

Figure 4.2

Vous allez maintenant attribuer des couleurs aux pays en fonction du continent auxquels ils appartiennent. Pour cela, vous allez utiliser l'un des nombreux modèles de couleurs disponibles dans la librairie D3[2]. Prenons par exemple le `schemeCategory10`. Pour l'utiliser, il vous faut, avant la boucle, créer une échelle ordinale (échelle de valeurs discrètes) de couleurs prenant en paramètre le modèle,

```
var colorScale = d3.scaleOrdinal(d3.schemeCategory10);
```

puis utiliser cette échelle pour la propriété `fill` de vos triangles, en passant en paramètres le continent du pays correspondant :

```
tr.attr("fill", colorScale(data[i]["continent"]));
```

Vous obtenez ainsi l'image de la figure 4.2.

Pour finir, vous allez attribuer une opacité de 0.5 à vos triangles de façon à mieux voir les éléments superposés. Pour cela, ajoutez l'attribut `tr.attr("fill-opacity", 0.5);`. Cela vous donne le rendu de la figure 5.

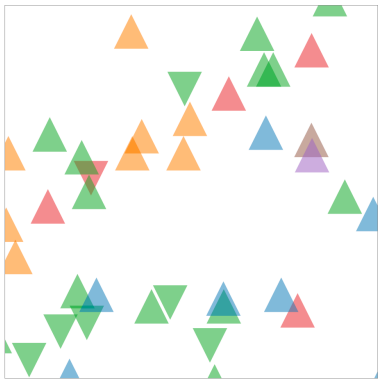

Figure 5

[2] https://github.com/d3/d3-scale-chromatic

En termes de sémiologie, vous avez donc produit un nuage de points dans lequel chaque point (implantation ponctuelle) représente un pays avec les variables visuelles suivantes :

- Position `x` du point : nombre de cas de COVID-19 dans le pays,
- Position `y` du point : nombre de morts liés à la COVID-19 dans le pays,
- Orientation de la forme du point : présence ou non de nouveaux cas de COVID-19 dans le pays,
- Teinte de la forme : continent du pays.

Votre nuage de points permet donc de visualiser 4 attributs des données : `total_cases`, `total_deaths`, `new_cases` et `continent`. Notez que la forme en elle-même du point (un triangle) ne représente rien, c'est son orientation qui code la donnée `new_cases`.

Exercice 2

Dans cet exercice, nous allons voir comment sélectionner plusieurs triangles se trouvant dans un rectangle tracé à l'aide de la souris. Pour cela, nous allons utiliser la fonctionnalité *brush* de D3 qui permet d'effectuer le tracé d'un rectangle avec la souris et renvoie ses coordonnées. Puis nous sélectionnerons les triangles qui se trouvent à l'intérieur.

Commencez par supprimer le zoom de votre visualisation.

Ensuite, afin de pouvoir parcourir les triangles, il vous faut les stocker dans un tableau. Juste avant la boucle de création des triangles, déclarez un tableau vide (`var triangles = [];`), puis, dans la boucle, ajoutez les `tr` dans ce tableau au fur et à mesure que vous les créez (`triangles[i]=tr;`). Vérifiez que vous obtenez bien un tableau d'objets à l'aide d'un `console.log`.

Ensuite, la fonctionnalité *brush* s'ajoute de la façon suivante

```
var b = d3.brush();
b.on("brush", function({selection}){
        // Le code placé ici est appelé quand l'utilisateur sélectionne un rectangle
});
gGlobal.call(b);
```

Testez. Normalement, vous pouvez créer un rectangle avec votre souris et le déplacer comme le montre la figure 6.1.

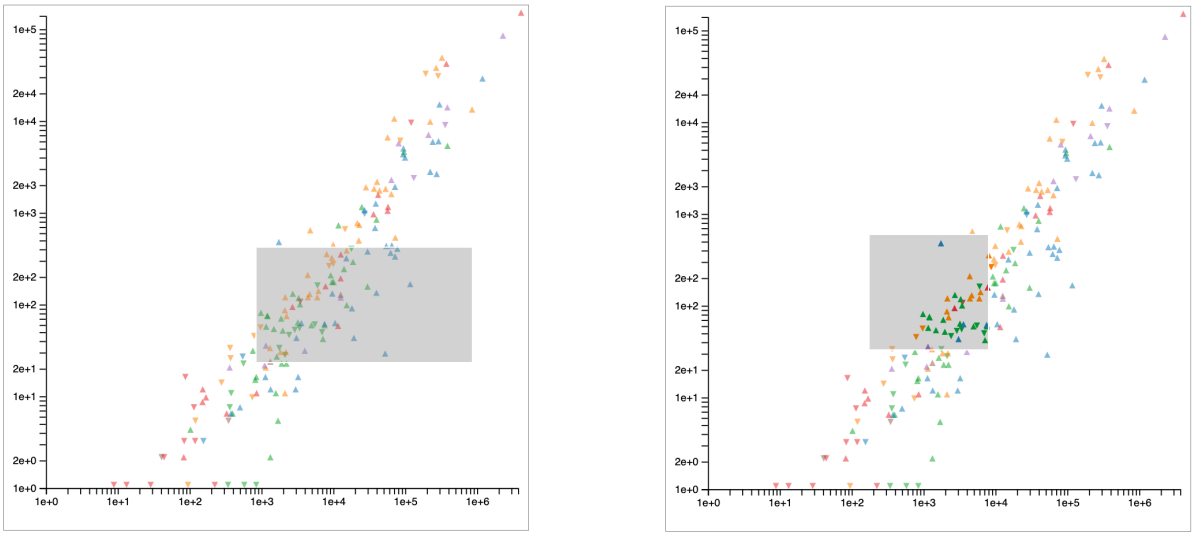

Figure 6.1 Figure 6.2

Il reste maintenant à remplir la fonction. Afin de récupérer les éléments sélectionnés, la fonction *brush* nous donne les coordonnées du rectangle de la sélection :

- Position x du coin en haut à gauche : selection[0][0],
- Position y du coin en haut à gauche : selection[0][1],
- Position x du coin en bas à droite : selection[1][0],
- Position y du coin en bas à droite : selection[1][1].

Comme exemple, nous allons simplement modifier l'opacité des triangles sélectionnés. Voici le code correspondant :

```
for(i=0;i<data.length;i++){
       if(data[i]["total_cases"]>0 &&
          data[i]["total_deaths"]>0){
             var pos_x = scaleX(data[i]["total_cases"])+40;
             var pos_y = scaleY(data[i]["total_deaths"])+10;
             if( pos_x >= selection[0][0] &&
                 pos_x <= selection[1][0] &&
                 pos_y >= selection[0][1] &&
                 pos_y <= selection[1][1] ){
                  triangles[i].attr("fill-opacity", 1);
             }
             else {
                  triangles[i].attr("fill-opacity", 0.5);
             }
       }
}
```

Pour chaque pays (ligne 1) ce code commence par vérifier que les valeurs `total_cases` et `total_deaths` ne sont pas nulles (lignes 2 et 3). En effet, rappelez-vous, nous n'avons pas créé les triangles quand c'était le cas. Ensuite, il récupère les coordonnées du triangle et les met dans les variables pos_x et pos_y (lignes 4 et 5). L'instruction conditionnelle suivante (lignes 6, 7, 8 et 9) regarde si les positions du triangle sont comprises dans le rectangle de la sélection. Si c'est le cas, l'opacité du triangle est passée à 1 (ligne 10), sinon, elle est passée à 0.5 (ligne 13).

Testez ce code, vous devriez obtenir une image comme celle de la figure 6.2.

Exercice 3 (entrainement)

Reprenez votre code de façon à afficher sur les axes d'autres attributs. Vous pouvez par exemple comparer le nombre de morts (`total_deaths`) avec le pourcentage de personnes de plus de 70 ans (`aged_70_older`) ou avec l'espérance de vie (`life_expectancy`).

Conclusion

Dans ce TP, nous avons réutilisé les notions abordées dans le TP précédent afin de dessiner un nuage de points sur des données réelles.

Nous avons aussi vu comment sélectionner un ensemble d'objets graphiques à l'aide de la fonction *brush*. La sélection est souvent employée en visualisation, elle permet notamment :

- De lancer des calculs sur une sous partie des données (on pourrait par exemple ajouter un `alert` affichant la moyenne du nombre de lit d'hôpital par millier dans l'exemple de l'exercice 2, nous avons toutes les données pour le faire),
- Mettre en surbrillance les éléments sélectionnés dans d'autres vues (nous en verrons des exemples en M2)
- Afficher le détail d'une liste d'éléments,
- …

Dans le prochain TP, nous verrons comment la sélection peut être utile pour faciliter la lecture de coordonnées parallèles.