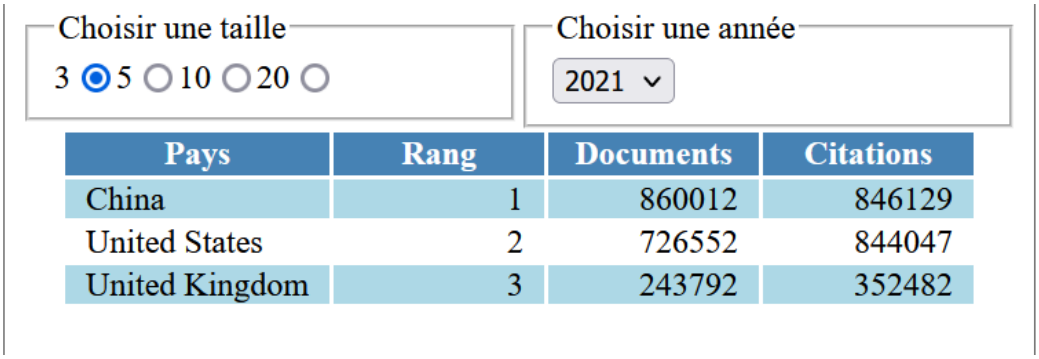
# TD n°2 – D3.js

## Contexte 1 : Données de production scientifique (exercice très guidé !)

L'objectif est de créer une page web synthétisant les productions scientifiques par pays avec une interaction sur le nombre de pays affiché. Voilà ce que vous obtiendrez à la fin de l'exercice :




| Pays | Rang | Documents | Citations |
|---|---|---|---|
| China | 1 | 860012 | 846129 |
| United States | 2 | 726552 | 844047 |
| United Kingdom | 3 | 243792 | 352482 |

Le fichier csv est récupérable sur Updago `scimagojr.csv` ou bien à l'adresse suivante :
https://francoisgarnier.github.io/scimagojr.csv

| TRAVAIL A FAIRE | |
|---|---|
| **1** | Créer le fichier `scientifique.html`. Le tableau sera visible dans la balise `<div>` que l'on identifiera avec l'id `top`. L'autre balise `<div>` identifiée par l'id `nuage` nous servira pour réaliser le travail à faire par la suite. Le fichier CSS `scientifique.css`, sera étudié plus tard. Nous chargeons la dernière version de la librairie d3js (version 7 juin 2021) :<br>`<!DOCTYPE html>`<br>`<html>`<br>`  <head>`<br>`  <meta charset = "UTF-8">`<br>`  <script src="https://d3js.org/d3.v7.min.js"></script>`<br>`  <link rel="stylesheet" href="scientifique.css">`<br>`  </head>`<br>`  <body>`<br>`  <div id = "top"></div>`<br>`  <div id = "nuage"></div>`<br>`  <script src = "scientifique.js"></script>`<br>`  </body>`<br>`</html>` |
| **2** | Créer le fichier `scientifique.js`. Tester l'appel de la librairie d3.js avec ce code :<br>`d3.select("head").append("title").html("d3js fonctionne !");`<br><br>Normalement votre page web fait apparaître un titre !<br><br>Ajouter un titre en h2 sur votre page html : « TOP des productions scientifiques » |
| **3** | Ajouter un bouton radio via le js qui nous permettra de choisir le nombre de pays à afficher :<br>`var choix_taille = d3.select("#top").append("form").attr("class", "choix").append("fieldset");`<br>`choix_taille.append("legend").html("Choisir une taille");`<br>`choix_taille`<br>`  .selectAll("input")`<br>`  .data([3, 5, 10, 20])` |

| | |
|---|---|
| | `.enter()`<br>`.append("label")`<br>`.text(function(d) {return d;})`<br>`.insert("input")`<br>`.attr("type", "radio")`<br>`.attr("name", "taille")`<br>`.attr("value", function(d) {return d;})`<br>`.property("checked", function(d) {return d == 5;})`<br>`.on("change", function() { console.log(this.value) }) ;` |
| **4** | Compléter le script js en faisant une liste déroulante mais vide. Les options de choix des années seront ajoutées plus tard, lorsque nous aurons chargé les données :<br><br>`var choix_annee = d3.select("#top").append("form").attr("class", "choix")`<br>`.append("fieldset");`<br>`choix_annee.append("legend").html("Choisir une année");`<br>`choix_annee.append("select").attr("name", "annee").attr("id", "choix_annee");` |
| **5** | Il est plus simple de créer la structure du tableau à remplir en amont, comme ci-dessous. La `table` HTML sera donc en deux parties : le nom des variables dans `thead` et les données dans `tbody`<br><br>`d3.select("#top").append("table")`<br>`.append("thead")`<br>`.append("tr")`<br>`.selectAll("th")`<br>`.data(["Pays", "Rang", "Documents", "Citations"])`<br>`.enter()`<br>`.append("th")`<br>`.html(function(d) {return d;});`<br>`d3.select("#top").select("table")`<br>`.append("tbody")`<br>`.attr("id", "table_top");` |
| **6** | Nous allons utiliser des fonctions qui s'avèreront utiles :<br><br>a) **Génération d'une cellule** :<br><br>Une cellule d'un tableau est dans une balise `<td>`. Cette fonction prend donc une valeur en paramètre et renvoie une chaîne de caractère ajoutant `"<td>"` devant et `"</td>"` derrière. Pour avoir un affichage plus agréable, on ajoute `"class='texte'"` pour une valeur chaîne de caractère et `"class='nombre'"` pour une valeur numérique :<br><br>`var generateCell = function(d) {`<br>`return "<td class='" + (isNaN(parseInt(d)) ? "texte" : "nombre") + "'>" + d + "</td>";`<br>`} ;`<br><br>b) **Génération d'une ligne** :<br><br>On utilise la fonction précédente pour créer une ligne à partir d'un élément du jeu de données, en indiquant donc le pays, le rang, le nombre de documents et le nombre de citations :<br>`var generateRow = function(d) {`<br>`return generateCell(d.Country) + generateCell(d.Rank) + generateCell(d.Documents) + generateCell(d.Citations)} ;` |

**7**

La fonction `d3.csv()` permet de récupérer des données stockées dans un fichier texte, avec comme séparateur une virgule. Il existe bien évidemment d'autres fonctions pour d'autres types de données textes. Le premier paramètre est l'adresse du fichier stockant les données. Le deuxième paramètre permet de définir une fonction à appliquer à chaque élément, souvent utilisée pour faire des transformations sur les données (ici, conversion en numérique de certaines variables). Sur l'objet créé par `d3.csv()`, on peut appliquer la fonction `.then()` qui, elle, applique la fonction définie en paramètre sur les données une fois celles-ci chargées. Nous verrons les 4 étapes, indiquées par les commentaires, par la suite.

```
d3.csv(
    "https://francoisgarnier.github.io/scimagojr.csv",
    function (d) {
        return {
            Year: parseInt(d.Year),
            Rank: parseInt(d.Rank),
            Country: d.Country,
            Documents: parseInt(d.Documents),
            Citations: parseInt(d.Citations),
            Hindex: parseInt(d["H index"])
        }
    })
    .then(function(data) {
        // Mise à jour du choix des années

        // Remplissage du tableau

        // Changement de taille

        // Changement d'année

    }) ;
```

**8**

Mettre à jour la liste des années du `select` nommé `choix_annee`. Le code suivant doit se placer juste après la première ligne de commentaire :

```
d3.select("#choix_annee").selectAll("option")
    .data(Array.from(new Set(data.map(function(d) {return d.Year;}))))
    .enter()
    .append("option")
    .attr("value", function(d) {return d;})
    .html(function(d) {return d;});
```

**9**

Remplir le tableau avec les données. Pour cela, nous sélectionnons déjà les données de l'année spécifiée (par défaut, c'est la dernière). Toutes les données seront toujours présentes, mais seules les 3, 5, 10 ou 20 premières seront visibles, en fonction du choix de l'utilisateur. C'est avec le `display` que l'on met à `none` si le rang est supérieur à la taille, ou à `table-row` sinon, que l'on peut limiter ce qui est visible par l'utilisateur. Dans la fonction passée en paramètre de `.style()`, le premier paramètre est donc les données et le second sa position dans le tableau des données. Puisque les données sont ordonnées dans le fichier d'origine, l'élément à la position `i` est au rang `i+1`.

```
var data_annee = d3.filter(data, function(d) {return d.Year ==
d3.select("#choix_annee").node().value;});
d3.select("#table_top").selectAll("tr")
    .data(data_annee)
    .enter()
    .append("tr")
    .style("display", function(d, i) {
```

```
        return (i+1) >
parseInt(d3.select('input[name="taille"]:checked').node().value) ?
"none" : "table-row";
    })
    .html(function(d) { return generateRow(d); }) ;
```

**10** On ajoute un suivi du changement de `choix_annee`, pour *cacher* ou non les lignes inutiles pour nous :

```
choix_taille.on("change", function() {

    d3.select("#table_top").selectAll("tr")
        .style("display", function(d, i) {
            return (i+1) >
parseInt(d3.select('input[name="taille"]:checked').node().value) ?
"none" : "table-row";
        })
}) ;
```

**11** Lorsqu'on change d'années, on doit récupérer les nouvelles données. Le travail est donc un peu plus conséquent :

```
choix_annee.on("change", function() {
    data_annee = d3.filter(data, function(d) { return d.Year ==
d3.select("#choix_annee").node().value});
    d3.select("#table_top").html("")
        .selectAll("tr")
        .data(data_annee)
        .enter()
        .append("tr")
        .style("display", function(d, i) {
            return (i+1) >
parseInt(d3.select('input[name="taille"]:checked').node().value) ?
"none" : "table-row";
        })
        .html(function(d) { return generateRow(d); });
});
```

**12** Enfin, pour finir, voici une proposition de fichier css ( à modifier si vous le souhaitez) :

```
.choix {
    width: 50%;
    vertical-align: top;
    display: inline-block;
}

#top table {
    margin: 0 auto;
}

#top tr th {
    text-align: center;
    min-width: 100px;
    background-color: steelblue;
    color: white;
}
#top td{
    padding: 0 10px;
}
#top td.texte {
    text-align: left;
}
```

| | |
|---|---|
| | `#top td.nombre {`<br>`  text-align: right;`<br>`}`<br>`#top tbody tr:nth-child(odd) {`<br>`  background-color: lightblue;`<br>`}`<br><br>**Tester l'ensemble de l'application web !**<br><br>**BONUS : Faites évoluer votre application en rajoutant une colonne pour vous approprier complètement le code (ex : la région)** |
| **13** | Faire le nuage de points entre le nombre de documents produits et le nombre moyen de citations pour la dernière année disponible, pour chaque pays, avec les contraintes suivantes :<br>• couleur (entre du vert pour le 1er et du rouge pour le dernier) en fonction de son rang<br>• taille en fonction du *H-index*<br>• lignes de références au niveau des moyennes de chaque variable<br>• axes logarithmiques<br>• nom de chaque pays indiqué lorsque la souris passe dessus<br><br>*Nuage de points avec D3.js :*<br>*http://kaisersly.github.io/scottmurray-d3-fr/14-faire-un-nuage-de-points.html* |

## Contexte 2 : AirBnB (un peu moins guidé !)

Le jeu de données concerne la répartition des logements en location par **AirBnB** sur Paris le 6 juin 2022, fichier csv récupérable sur Updago ou directement avec l'URL :
http://data.insideairbnb.com/france/ile-de-france/paris/2022-06-06/visualisations/listings.csv

L'objectif de ce contexte est de réaliser une page web qui affiche dans un tableau HTML, le nombre de logements par type de logement avec le prix moyen :

| TRAVAIL A FAIRE | |
|---|---|
| **1** | Nous allons commencer par travailler sur des données fictives.<br><br>Ecrire les 3 fichiers nécessaires à la réalisation de la page web :<br><br>- un fichier `airbnb_ex1.js` le code suivant et le comprendre :<br>`//Données statiques : type = logement complet, chambre privée, chambre partagée`<br>`donnees = [`<br>`  { type: "Entire home/apt", count: 35185, price: 106 },`<br>`  { type: "Private room", count:  5827, price:  56 },`<br>`  { type: "Shared room", count:  464, price:  40 },`<br>`  { type: "Other type", count: 10, price: 200}`<br>`];`<br>`var type_modalites = donnees.map(function(d) { return d.type; });`<br>`var prices = donnees.map(function(d) { return d.price; });`<br>`var price_min = d3.min(prices);`<br>`var price_max = d3.max(prices);`<br><br>`var qt2num = d3.scaleLinear()`<br>`  .domain([price_min, price_max])`<br>`  .range([0, 100]);`<br>`var qt2col = d3.scaleLinear()`<br>`  .domain([price_min, price_max])`<br>`  .range(["lightgreen", "steelblue"]);`<br>`var ql2num = d3.scaleBand()` |

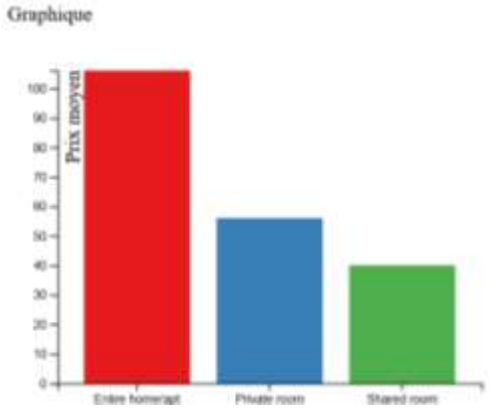
```
    .domain(type_modalites)
    .range([0, 100]);
var ql2col = d3.scaleOrdinal(d3["schemeSet1"])
    .domain(type_modalites);

var tbody = d3.select("#tab").selectAll("tr")
  .data(donnees)
  .enter().append("tr")
  .html(function(d) {
    var chaine = "<td>" + d.type + "</td>" +
        "<td>" + ql2num(d.type) + "</td>" +
        "<td style='background-color:" + ql2col(d.type) + "'></td><td></td>" +
        "<td>" + d.price + "</td>" +
        "<td>" + qt2num(d.price) + "</td>" +
        "<td style='background-color:" + qt2col(d.price) + "'></td>";
    return chaine;
  });
```

- un fichier `airbnb_ex1.css qui` ne contiendra que la ligne suivante :
  `td { text-align: center;}`

- un fichier `airbnb_ex1.HTML` à écrire qui contiendra un tableau à afficher avec les données statiques du fichier js (étudier le fichier js). Le résultat devra donner cette visualisation :

| | **Type** (qualitative) | | **Price** (quantitative) | | |
|---|---|---|---|---|---|
| **original** | **en numérique** | **en couleur** | **original** | **numérique** | **couleur** |
| Entire home/apt | 0 | | 106 | 41.25 | |
| Private room | 25 | | 56 | 10 | |
| Shared room | 50 | | 40 | 0 | |
| Other type | 75 | | 200 | 100 | |

**2** Créer le graphique suivant :




Pour cela, écrire les 2 fichiers suivants :

- un fichier html contenant les références au fichier `airbnb_ex2.js` ainsi que les balises :

```
<p>Graphique</p>
<div id="graph"></div>
```

- un fichier `airbnb_ex2.js` :

```
// données statiques
donnees = [
  { type: "Entire home/apt", count: 35185, price: 106 },
  { type:    "Private room", count:  5827, price:  56 },
  { type:     "Shared room", count:   464, price:  40 }
];

// Liste des modalités de la variable type
var type_modalites = donnees.map(function(d) { return d.type; });
// Prix (moyen) maximum
var prix_max = d3.max(donnees, function(d) { return d.price; });

// Définition des marges et de la taille du graphique
var marges = {haut: 20, droit: 20, bas: 30, gauche: 40},
    largeurTotale = 400,
    hauteurTotale = 300,
    largeurInterne = largeurTotale - marges.gauche - marges.droit,
    hauteurInterne = hauteurTotale - marges.haut - marges.bas;

// Echelle pour les prix sur l'axe Y
var echelleY = d3.scaleLinear()
    .domain([0, prix_max])
    .range([hauteurInterne, 0]);
// Echelle pour le type sur l'axe X
var echelleX = d3.scaleBand()
    .domain(type_modalites)
    .range([0, largeurInterne])
  .padding(0.2);
// Echelle pour le type affectant une couleur automatique à chaque type
var echelleCouleur = d3.scaleOrdinal(d3["schemeSet1"])
    .domain(type_modalites);

// Création de l'axe X
var axeX = d3.axisBottom()
    .scale(echelleX);

// Création de l'axe Y
var axeY = d3.axisLeft()
    .scale(echelleY);

// Création du graphique
var graphique = d3.select("#graph").append("svg")
    .attr("width", largeurTotale)
    .attr("height", hauteurTotale)
  .append("g")
    .attr("transform", "translate(" + marges.gauche + "," + marges.haut
+ ")");

// Ajout de l'axe X au graphique
graphique.append("g")
    .attr("class", "x axis")
    .attr("transform", "translate(0," + hauteurInterne + ")")
  .call(axeX);

// Ajout de l'axe Y au graphique
graphique.append("g")
    .attr("class", "y axis")
  .call(axeY);

graphique.append("text")
    .attr("transform", "rotate(-90)")
    .attr("y", 6)
    .attr("dy", ".71em")
    .style("text-anchor", "end")
    .text("Prix moyen");
```

```
// Ajout d'une barre pour chaque type de logement, avec une taille fonction du prix moyen
graphique.selectAll(".bar")
  .data(donnees)
  .enter()
  .append("rect")
  .attr("class", "bar")
  .attr("x", function(d) { return echelleX(d.type); })
  .attr("width", echelleX.bandwidth())
  .attr("y", function(d) { return echelleY(d.price); })
  .attr("height", function(d) { return hauteurInterne - echelleY(d.price); })
  .style("fill", function(d) { return echelleCouleur(d.type); });
```

**3**

En vous référant au cours, écrire un script javascript `airbnb_ex3.js` permettant de lire un fichier csv externe ci-dessous avec D3.js pour que les données des scripts précédents soient dynamiques.
url :http://data.insideairbnb.com/france/ile-de-france/paris/2022-06-06/visualisations/listings.csv

Il faudra bien évidement travailler les données récupérées pour obtenir notre tableau de données intial (`donnees` dans le script précédent). Pour cela, il faudra notamment transtyper certaines données chaines en numérique (prix). Pour vous aider dans cette tâche, suivez les indications données dans ce lien :
https://observablehq.com/@fxjollois/initiation-a-d3js-lecture-de-donnees

L'objectif de ce script étant d'obtenir sur la même page web :

- Les deux tableaux des scripts précédents mais dynamiquement à partir du fichier csv.
- Trois autres indicateurs utiles pour un dirigeant d'Airbnb Paris (graphiques, tableaux dynamiques avec sélection (voir ex1), …). *Regarder les variables de départ dans le fichier csv*
- Au moins un évènement à gérer sur la page html de départ.

La page html regroupant ce tableau de bord devra être ergonomique et **interactive**. Tout doit être consigné en javascript, la page html de départ ne contiendra aucune balise.

Bon courage !