# Corrigé du TD – D3.js avancé

*Objectif : la séance a pour objectif de mettre en place une carte interactive des gares SNCF en France affichant des informations sur leur fréquentation.*

**Les jeux de données :**

Départements.json : https://raw.githubusercontent.com/Lawiss/d3js/master/departments.json

gares.json : https://raw.githubusercontent.com/Lawiss/d3js/master/gares.json

---

Rendez vous sur le site (trame de départ) : https://codepen.io/idriss_/pen/mzzdMr

Pour la viédo du TD corrigé: ttps://youtu.be/BvQrQADU6-I

Dans la partie HTML :

1. Insérer une balise script dans le header du code HTML pour insérer le lien de la bibliothéque d3.js : *https://d3js.org/d3.v5.min.js*

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

2. Ajouter dans le corps du fichier une balise div avec comme id: *carte*

```
<div id="carte">

</div>
```

A présent, on ne travaillera que dans la partie Javascript :

3. Créer un élément SVG à l'aide de D3 avec les valeurs de width et height que vous aurez définis au préalable.

```
//Configuration de la taille du bloc qui va accueillir les vecteurs
```

```
graphiques
const width = 720, height = 720;

//Création du bloc svg qui contiendra les départements (polygones) et
les points (circles) une fois projetés
const svg = d3.select('#carte').append("svg")
    .attr("id", "svg")
    .attr("width", width)
    .attr("height", height)
```

Partie 1 : Affichage de la carte de France

1. Le code suivant définit les objets nécessaires pour configurer la projection géographique de nos données sur le plan 2D de notre page. A l'aide des méthodes .center(), définissez le centre de la projection à la latitude et la longitude suivante : [2.454071, 46.279229] afin que notre projection soit centrée sur la France. Vous pouvez également régler l'échelle avec la méthode scale (*nous recommandons 3600*)

```
//Configuration de la projection utilisée
const projection = d3.geoConicConformal()
    .center([2.454071, 46.279229])
    .scale(3600)
    .translate([width / 2, height / 2])
    ;

const pathD3 = d3.geoPath();
```

2. Assigner la projection à l'objet pathD3 avec la méthode du path : projection()

```
pathD3.projection(projection);
```

3. Ajouter un élément *g* (groupe) au svg créé précédemment : append()

```
//création d'un groupe ("g") dans le bloc svg qui contiendra les
départements
const deps = svg.append("g");
```

4. Charger les données geojson du fichier departement.json : d3.json() et dessiner les données des départements. Chaque départements est défini dans le JSON par un path (polygone complexe) en coordonnées géographique. VOIR TUTO PPT.
   *A l'aide de la fonction pathD3 définie précédemment, on peut projeter ces coordonnées dans un plan 2D pour pouvoir ensuite les attribuer aux éléments path de notre SVG.*

```
//Téléchargement des départements et création+projection des polygones
d3.json('https://raw.githubusercontent.com/Lawiss/d3js/master/department
s.json').then(function(geojson) {


    deps.selectAll("path")
        .data(geojson.features)
        .enter()
        .append("path")
        .attr("d", pathD3)
        .attr("fill","#fad390")
        .attr("stroke","black");
  });
```

5. Customiser le rendu de la carte avec les attributs : fill et stroke.

```
        .attr("fill","#fad390")
        .attr("stroke","black");
```

## Partie 2 : Affichage des gares sur la carte

Le fichier gares.json contient les coordonnées et informations de plusieurs milliers de gares françaises. Les informations de la gare sont contenues dans “properties” tandis que les coordonnées sont dans “geometry” puis “coordinates”.

1. Ajouter un élément *g* (groupe) au svg créé précédemment : append()

```
//création d'un groupe ("g") dans le bloc svg qui contiendra les gares
circles = svg.append("g");
```

2. Charger les données geoJSON : gares.json.

```
//Téléchargement des gares, traitement pour transformer le JSON en
geoJSON (facultatif) et projection.
d3.json("https://raw.githubusercontent.com/Lawiss/d3js/master/gares.json
").then(function(gares){
```

3. Projeter ces données. Utilisez cette fois la méthode projection() définies précédemment pour obtenir les coordonnées de chaque cercle. Le rayon d’un cercle est défini par l’attribut “r”.

```
  //PRojection des cercles, chaque bloc <circle> a un évenement qui
appelle la fonction showInfo ou hideInfo en fonction que l'on passe la
souris dessus ou pas
     circles.selectAll("circle")
            .data(collection.features)
            .enter()
            .append("circle")
            .attr("r", 2)
            .attr("cx", function(d) {return
projection(d.geometry.coordinates)[0]})
            .attr("cy", function(d) {return
projection(d.geometry.coordinates)[1]})
            .on("click",openModalWindow);
```

Partie 3 : Affichage des informations au survol

1. Insérer un élément div de classe **tooltip** dans le corps HTML en passant par D3.js et stockez-le dans une variable nommée div. Utilisez la méthode style() pour rendre invisible ce div.

```
/Création du bloc qui va contenir les infos
var div = d3.select("body").append("div")
    .attr("class", "tooltip")
    .style("opacity",0)
    .style("display", "none");
```

2. Écrire une fonction qui prend en entrée un tableau contenant la fréquentation d'une gare sur trois ans et retournant la moyenne de la fréquentation.

```
function moyenneFreq(freqs){
  somme=0
  for(i=0;i<3;i++){
      somme=somme+freqs[i]
  }

  return somme/3
}
```

3. Créer une fonction showInfo(gare) qui prend en entrée un objet gare du geoJSON, qui calcule la moyenne de sa fréquentation et qui remplit l'élément div créé précédemment avec le nom de la gare, sa fréquentation moyenne et enfin qui fait apparaître l'élément. Pour vous aider :

```
//Fonction qui fait apparaître le bloc contenant les infos au dessus de
la souris
function showInfo(d){
    div.style("display","block")
```

```
    div.transition()
    .duration(200)
    .style("opacity", .9);
    div.html("<b>Nom de la gare : </b>" + d.properties.LIBELLE_GARE +
".<br/>"
      +  "<b>Fréquentation moyenne : </b>" +
moyenneFreq(d.properties.VOYAGEURS))
    .style("left", (d3.event.pageX + 30) + "px")
    .style("top", (d3.event.pageY - 30) + "px")


}
```

4. Créer une autre fonction hideInfo() qui fait disparaître l'élément div.

```
//fonction qui fait disparaître les infos
function hideInfo(d){
    div.style("opacity", 0);
    div.html("")
    div.style("display","none")
}
```

5. Dans l'instruction qui affiche les points ajouter deux évènements de type: mouseover qui appelle showInfo et mouseout qui appelle hideInfo().

```
  //PRojection des cercles, chaque bloc <circle> a un évenement qui
appelle la fonction showInfo ou hideInfo en fonction que l'on passe la
souris dessus ou pas
     circles.selectAll("circle")
            .data(collection.features)
            .enter()
            .append("circle")
            .attr("r", 2)
```

```
            .attr("cx", function(d) {return
projection(d.geometry.coordinates)[0]})
            .attr("cy", function(d) {return
projection(d.geometry.coordinates)[1]})
            .on("mouseover",function(d){showInfo(d)})
            .on("mouseout",function(d){hideInfo(d)})
            .on("click",openModalWindow);
});
```

6. Modifier votre fonction showInfo() pour positionner l'élément div par rapport à la position du curseur. Pour vous aider:

```
//Fonction qui fait apparraitre le bloc contenant les finos au dessus de
la souris
function showInfo(d){
    div.style("display","block")
    div.transition()
    .duration(200)
    .style("opacity", .9);
    div.html("<b>Nom de la gare : </b>" + d.properties.LIBELLE_GARE +
".<br/>"
      +  "<b>Fréquentation moyenne: </b>" +
moyenneFreq(d.properties.VOYAGEURS))
    .style("left", (d3.event.pageX + 30) + "px")
    .style("top", (d3.event.pageY - 30) + "px")


}
```

- **Bonus** : ajoutez la possibilité de zoomer sur la carte en appelant la fonction d3.zoom.on("zoom",function) à l'aide de la méthode call() au moment de la création du SVG.

Function est votre fonction callback qui va venir modifier les départements et les cercles des gares. Pour vous aider :

## Partie 4 : Affichage d'une fenêtre modale au clic

Dans cette partie vous allez manipuler les éléments suivants qui se trouvent déjà dans trame HTML :

```
<div class="modal-contenu">
      <span class="close">&times;</span>

      <div id="nom-gare"></div>
      <div id="conteneur-svg-modal"></div>
</div>
```

1. Créer une fonction openModalWindow() qui prends en paramètre un objet JSON gare. Cette fonction devra changer le style de l'élément ayant pour id "modal" pour le faire apparaître. Elle devra également modifier l'élément ayant pour id "nom-gare" pour y insérer le nom de la gare.

```
//sélection de notre élément #modal pour pouvoir le réutiliser ensuite
var modal = d3.select("#modal");

//Fonction qui ouvre la fenêtre modale
function openModalWindow(d){
      modal.select("#nom-gare")
      .text(d.properties.LIBELLE_GARE)

      modal.style("display","block")

}
```

2. Créer une fonction closeModalWindow() qui changera le style de de l'élément ayant pour id "modal" pour le faire disparaître.

```
function closeModalWindow(){

      modal.style("display","none");
      modal.select("#conteneur-svg-modal").text("");
}
```

3. Ajouter un événement au clic sur l'élément ayant pour classe "close" appelant la fonction closeModalWindow().

```
modal.select(".close").on("click",closeModalWindow)
```

4. Créer une fonction drawChart() qui prends pour paramètre un objet JSON gare et qui va créer un diagramme à barre représentant la fréquentation des voyageurs pour la gare sur les années 2014,2015 et 2016. Pour cela, définir un nouvel élément svg dans l'élément ayant pour id "conteneur-svg-modal". Utilisez les dimensions suivantes :

```
//Dimensions du SVG:
svgWidth=960;
svgHeight = 500;

//Les dimensions suivantes serviront pour définir la longueur des axes à
l'aide de la méthode range.
margin = { top: 50, right: 50, bottom: 30, left: 70 };
widthChart = svgWidth - margin.left - margin.right;
heightChart = svgHeight - margin.top - margin.bottom;

//Domaine de définition de l'axe x:
```

```
annees=["2014","2015","2016"]
```

a. Créer les échelles des axes : axe x avec une échelle d3.scaleBand() et axe y avec une échelle d3.scaleLinear(). N'oubliez pas de définir un range() et un domaine de définition avec domain().

```
function drawChart(d){
      //Annees va correspondre aux différentes valeurs prises sur l'axe
x
      annees=["2014","2015","2016"]

      //Je configure les dimensions en prenant une marge pour éviter que
les éléments, en particulier les axes, sortent du SVG et soient
invisibles
      svgWidth=960;
      svgHeight = 500;
      margin = { top: 50, right: 50, bottom: 30, left: 70 };
      widthChart = svgWidth - margin.left - margin.right;
      heightChart = svgHeight - margin.top - margin.bottom;

      //Création du SVG
      modalSVG = d3.select("#conteneur-svg-modal").append("svg")
      .attr("height",svgHeight)
      .attr("width",svgWidth);

      //max est la valeur maximale de l'axe x
      max=d3.max(d.properties.VOYAGEURS);

      //Je créé x et y qui sont les échelles et les domaines de
défnition  de mes deux axes
      x = d3.scaleBand().range([margin.left,
widthChart]).domain(["2014","2015","2016"]).padding("0.5");
      y = d3.scaleLinear().range([heightChart,
0]).domain([0,max+max*0.25]);
```

```
}
```

b. Utiliser les méthodes d3.axisBottom() et d3.axisLeft() pour créer les axes avec les échelles définies précédemment.

```
function drawChart(d){
      //Annees va correspondre aux différentes valeurs prises sur l'axe
x
      annees=["2014","2015","2016"]

      //Je configure les dimensions en prenant une marge pour éviter que
les éléments, en particulier les axes, sortent du SVG et soient
invisibles
      svgWidth=960;
      svgHeight = 500;
      margin = { top: 50, right: 50, bottom: 30, left: 70 };
      widthChart = svgWidth - margin.left - margin.right;
      heightChart = svgHeight - margin.top - margin.bottom;

      //Création du SVG
      modalSVG = d3.select("#conteneur-svg-modal").append("svg")
      .attr("height",svgHeight)
      .attr("width",svgWidth);

      //max est la valeur maximale de l'axe x
      max=d3.max(d.properties.VOYAGEURS);

      //Je créé x et y qui sont les échelles et les domaines de
défnition  de mes deux axes
      x = d3.scaleBand().range([margin.left,
widthChart]).domain(["2014","2015","2016"]).padding("0.5");
      y = d3.scaleLinear().range([heightChart,
```

```
0]).domain([0,max+max*0.25]);

      //Je construis mes axes
      var xAxis = d3.axisBottom(x);

      var yAxis = d3.axisLeft(y);





}
```

c. Ajoutez les axes au SVG en appliquant les transformations suivantes :

```
modalSVG.append("g")
      .attr("transform","translate(0,"+heightChart+")")
      .call(xAxis)

modalSVG.append("g")
      .call(yAxis)
      .attr("transform","translate("+margin.left+",0)")
```

```
function drawChart(d){
      //Annees va correspondre aux différentes valeurs prises sur l'axe
x
      annees=["2014","2015","2016"]

      //Je configure les dimensions en prenant une marge pour éviter que
les éléments, en particulier les axes, sortent du SVG et soient
invisibles
```

```
      svgWidth=960;
      svgHeight = 500;
      margin = { top: 50, right: 50, bottom: 30, left: 70 };
      widthChart = svgWidth - margin.left - margin.right;
      heightChart = svgHeight - margin.top - margin.bottom;

      //Création du SVG
      modalSVG = d3.select("#conteneur-svg-modal").append("svg")
      .attr("height",svgHeight)
      .attr("width",svgWidth);

      //max est la valeur maximale de l'axe x
      max=d3.max(d.properties.VOYAGEURS);

      //Je créé x et y qui sont les échelles et les domaines de
défnition  de mes deux axes
      x = d3.scaleBand().range([margin.left,
widthChart]).domain(["2014","2015","2016"]).padding("0.5");
      y = d3.scaleLinear().range([heightChart,
0]).domain([0,max+max*0.25]);

      //Je construis mes axes
      var xAxis = d3.axisBottom(x);

      var yAxis = d3.axisLeft(y);

      //J'ajoute à mon SVG l'axe x et le nom de l'axe
      modalSVG.append("g")
      .attr("transform","translate(0,"+heightChart+")")
      .call(xAxis)

      //J'ajoute ici l'axe y avec un petit texte
      modalSVG.append("g")
      .call(yAxis)
      .attr("transform","translate("+margin.left+",0)")
}
```

d. Créer et ajoutez au SVG les trois rectangles à partir des données contenues dans l'attribut .properties.VOYAGEURS. Pour positionner le rectangle et régler sa taille utilisez les axes x et y créés précédemment :

```
.attr("x",function(d,i){ return x(annees[i]);})
.attr("y",function(d,i){return y(d)})
.attr("height",function(d,i){return Math.abs(heightChart-y(d))})
```

```
function drawChart(d){
      //Annees va correspondre aux différentes valeurs prises sur l'axe x
      annees=["2014","2015","2016"]

      //Je configure les dimensions en prenant une marge pour éviter que
les éléments, en particulier les axes, sortent du SVG et soient
invisibles
      svgWidth=960;
      svgHeight = 500;
      margin = { top: 50, right: 50, bottom: 30, left: 70 };
      widthChart = svgWidth - margin.left - margin.right;
      heightChart = svgHeight - margin.top - margin.bottom;

      //Création du SVG
      modalSVG = d3.select("#conteneur-svg-modal").append("svg")
      .attr("height",svgHeight)
      .attr("width",svgWidth);

      //max est la valeur maximale de l'axe x
      max=d3.max(d.properties.VOYAGEURS);

      //Je créé x et y qui sont les échelles et les domaines de défnition
de mes deux axes
      x = d3.scaleBand().range([margin.left,
```

```
widthChart]).domain(["2014","2015","2016"]).padding("0.5");
      y = d3.scaleLinear().range([heightChart,
0]).domain([0,max+max*0.25]);

      //Je construis mes axes
      var xAxis = d3.axisBottom(x);

      var yAxis = d3.axisLeft(y);

      //J'ajoute à mon SVG l'axe x et le nom de l'axe
      modalSVG.append("g")
      .attr("transform","translate(0,"+heightChart+")")
      .call(xAxis)

      //J'ajoute ici l'axe y avec un petit texte
      modalSVG.append("g")
      .call(yAxis)
      .attr("transform","translate("+margin.left+",0)")

      //Ici je crée mes trois barres
      modalSVG.selectAll("rect")
      .data(d.properties.VOYAGEURS)
      .enter().append("rect")
      .attr("height",function(d,i){return Math.abs(heightChart-y(d))})
      .attr("width",x.bandwidth())
      .attr("x",function(d,i){ return x(annees[i]);})
      .attr("y",function(d,i){return y(d)})
      .attr("fill","#0a3d62")
      ;



}
```