Ajout d'un nouveau type de graphique

La première tâche nous ayant été confié fut de rajouter un nouveau type de graphique au projet déjà existant. Ce nouveau graphique a été un nuage de points. Pour cela, on a d'abord mise en place le nuage de point dans un fichier a part mais se basant sur le même format que le projet. Afin de pouvoir afficher le nuage de point sur la page web, il a fallu effectuer plusieurs étapes.

La première étape a été de mettre en place les échelles des différents axes pour le nuage de points. Pour cela on a utilisé une fonction de la bibliothèque D3 permettant de créer une échelle en fournissant l'étendue des valeurs, c'est-à-dire la valeur maximum et minimum des valeurs x et y présent dans l'ensemble des points afficher dans le nuage, et la taille souhaitait pour l'affichage des axes. Une fois ces échelles construites, il est nécessaire de créer les axes via la fonction axis() de D3.js permettant de créer un objet SVG comportant tous les éléments nécessaire à l'affiche dans la page HTML, c'est-à-dire la balise PATH permettant de dessiner le trait de l'axe et les balises affichant les valeurs à côté de l'axe. Afin de créer cet objet il faut lors de l'appel de la fonction fournir l'échelle correspondante afin de pouvoir afficher les bonnes valeurs pour l'axe et définir sa taille. Il est aussi possible de définir l'épaisseur souhaitait pour l'axe, la position de l'axe afin de choisir si l'axe doit être l'axe x ou y, et il est aussi possible de définir le nombre de valeurs maximal à afficher sur l'axe.

```
var echelleY = d3.scale.linear()
                    .domain([nuageRange[1][0],nuageRange[1][1]])
                    .range([maxy,miny]);

var echelleX = d3.scale.linear()
                    .domain([nuageRange[0][0],nuageRange[0][1]])
                    .range([0,axe]);

var xAxe = d3.svg.axis()
                .scale(echelleX)
                .tickSize(0.5*(cellSize/100))
                .ticks(4)
                .orient("bottom");

var yAxe = d3.svg.axis()
              .scale(echelleY)
              .tickSize(0.5*(cellSize/100))
              .ticks(4)
              .orient("left");
```

Une fois les axes créés, il a fallu mettre en place pour chaque cellule une balise qui recevra les différents éléments nécessaire à l'affichage du graphique dans la cellule. Pour cela, on ajoute à la zone dédié au SVG (balise SVG : zone comportant les cellules et les graphiques) autant de balise g (une balise permettant de regrouper différents objets tel que les axes et les points du nuage) qu'il y a de cellule présente. Pour chacune de ces balises, on lui définit ses dimensions afin de s'afficher correctement dans la cellule. On lui fournit aussi un ID afin de

pouvoir plus facilement accéder aux balises de chaque cellule lors de l'ajout des axes et des points du nuage. En plus de cela il faut aussi effectuer une translation à ces balises afin de pouvoir faire en sorte que les balises soient bien affichées à leur cellule correspondante. Pour cela lors de la création des cellules, un tableau comportant le centre de chaque cellule (c'est-à-dire son positionnement x et y dans la balise SVG) est alimenté. On ajoute alors l'attribut transform (attribut permettant de manipuler des éléments HTML dans l'espace 2D ou 3D) avec comme valeur translate et les valeurs x et y de chaque cellule à chaque balise g.

```
for (var indice = 0; indice < nbRows*nbColumns; indice++) {
        svg.append("g")
         .attr("id", "ggraph" + (indice + 1) )
         .attr("width", (width) + "px")
         .attr("height",(height) + "px")
         .attr("transform", "translate(" + (coordinatesArrayX[indice]) + ',' +
(coordinatesArrayY[indice]) + ')' );
                }
```

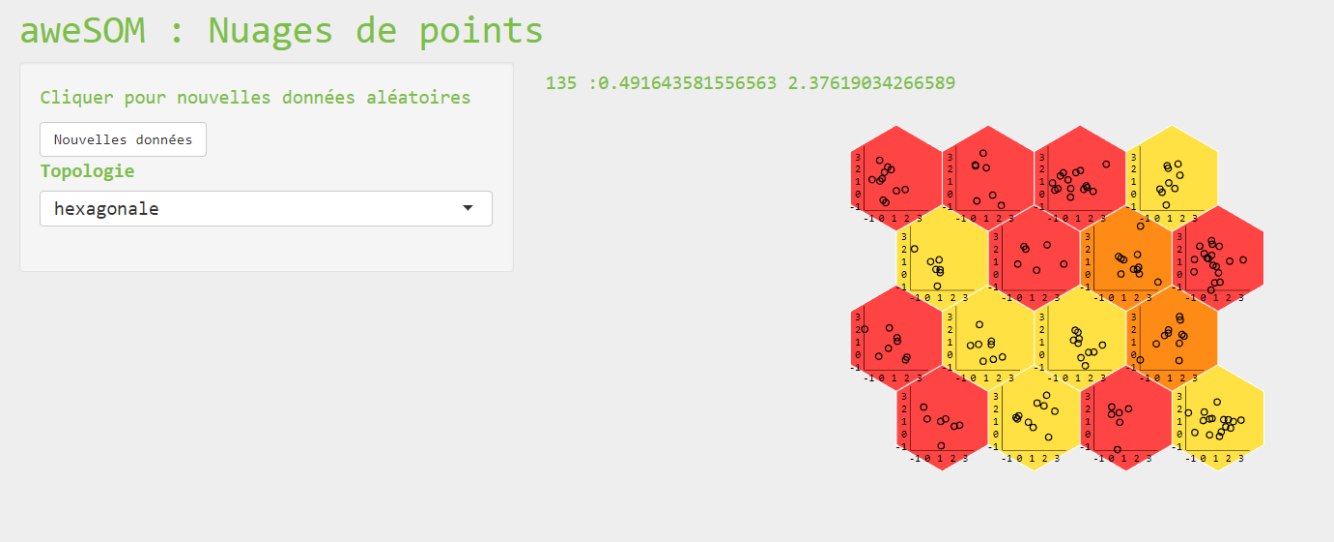
Une fois ces zones dédiées aux graphiques, il a fallu y ajouter les axes et les points du nuage. Pour cela, on parcourt l'ensemble des balises g créés précédemment et on ajoute une nouvelle balise g qui contiendra les éléments nécessaire à l'affichage de l'axe. Une fois cette balise créé, l'ajout des éléments de l'axe s'ajoute automatiquement en utilisant la fonction call() de D3 et en y fournissant en paramètre l'objet comportant les informations sur l'axe crée au début.

```
g.append("g")
  .attr("class","axis")
  .attr("font-size", cellSize/10 + "px")
  .attr("transform", "translate("+ 0 +","+ (axe + transpoX) +")" )
  .call(xAxe);
```

Une fois les axes ajoutés, il a fallu ajouter les points du nuage. Pour cela, les données relatives aux points étaient stockées dans un objet composé d'une liste. Chaque élément de cette liste étaient composé de l'ensemble des points à ajouter à la cellule selon l'indice de l'élément. Chaque point possédaient trois informations : son label, sa valeur x et sa valeur y. Afin de pouvoir ajouter les points à leur cellule attitrée, la méthode utilisée a été d'extraire l'élément comportant les points dans liste. Une fois cet élément récupéré, il a fallu le parcourir afin de récupérer chaque point devant être ajouter. A chaque points récupérer, on récupère la balise g de la cellule correspondante et on y ajoute une nouvelle balise qui comportera les informations du point. Cette balise est une balise circle permettant de dessiner un cercle dans le graphique. Il a été nécessaire de fournir différents attributs à cette balise afin de pouvoir la placer correctement dans le graphique. Il a fallu fournir sa dimension r afin de définir sa taille. Et il a fallu aussi fournir sa position x et y mais en les modifiant via les échelles produites précédemment afin qu'elle ne sorte pas de la zone définie.

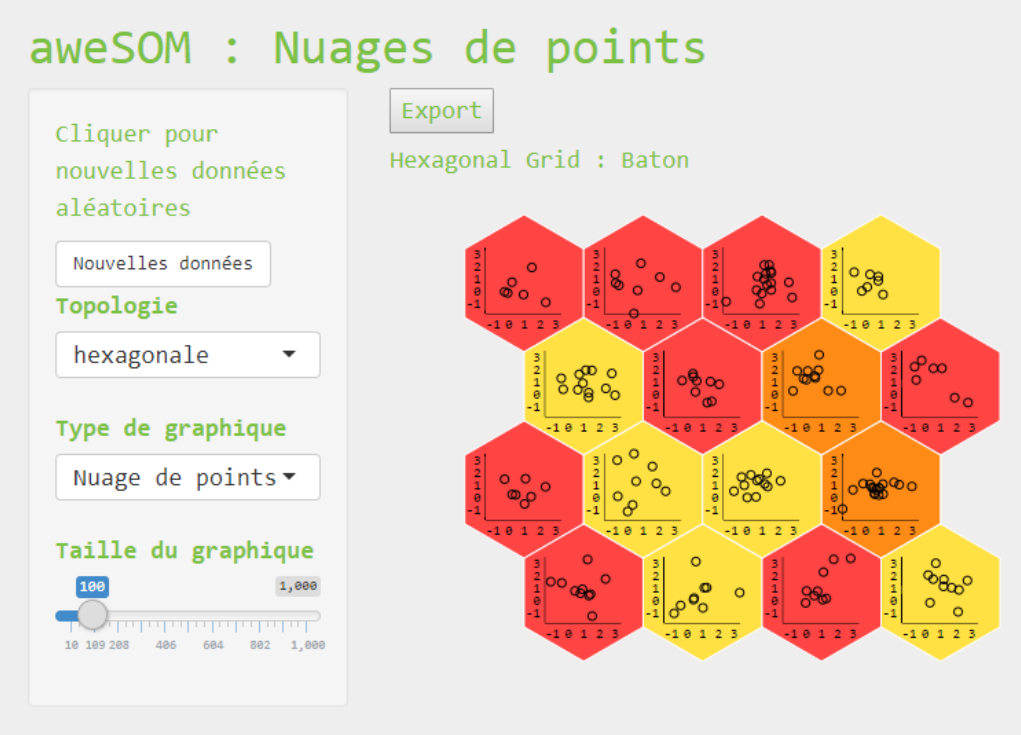
```
for (var indice=0; indice<nbRows*nbColumns; indice++){
    var dataCell=nuageData[indice+1];

        for(var j=0; j<dataCell.x.length; j++){
          var g=d3.select("#ggraph" + (indice+1));
          g.append("circle")
            .attr("cx",function(){
               return (echelleX(dataCell.x[j]));})
            .attr("cy",function(){
               return (echelleY(dataCell.y[j]));})
            .attr("r",function(){
               return 3*(cellSize/100);})
                 .style("stroke", "black")
                 .style("fill","none");}
}
```

Une fois ces différentes étapes effectuées, on obtient un graphique de type nuage de points dans chaque cellule. Pourtant ces étapes ne furent pas suffisantes, comme expliqué dans l'existant de ce projet, il y avait comme possibilité de choisir la forme des cellules (rectangulaire ou hexagonale). Hors lorsque le format choisi était en hexagonal, les valeurs des axes empiétés sur les cellules voisines.



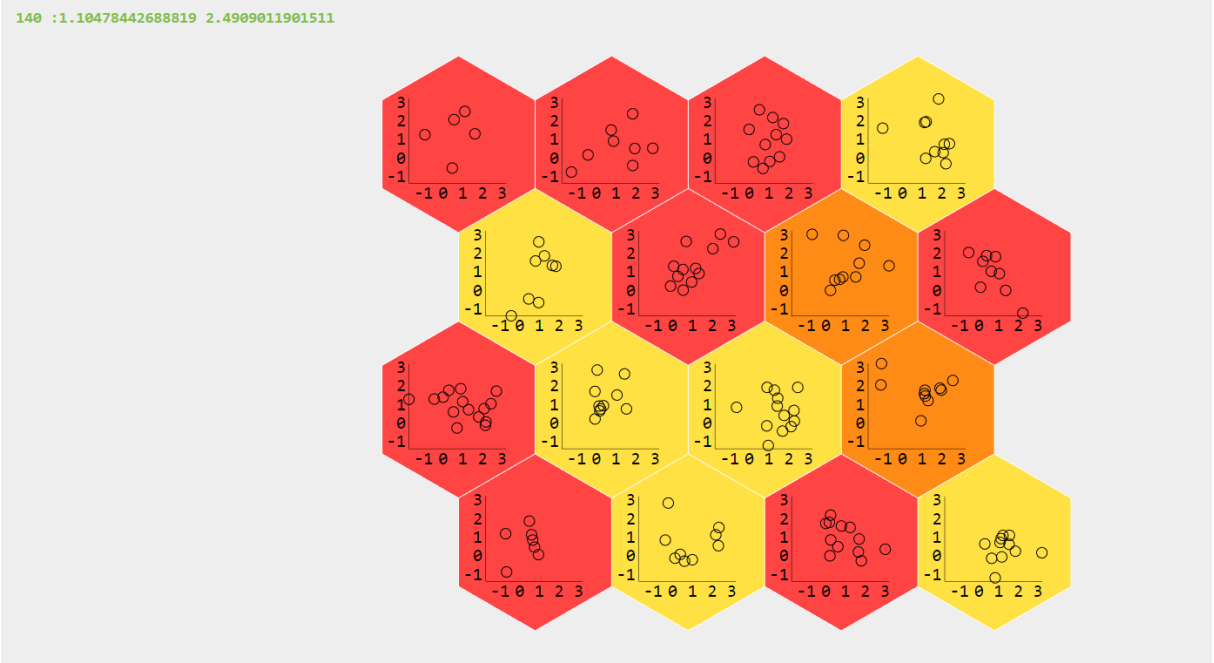
Afin de pallier à ce problème il a été nécessaire de faire des ajustements aux étapes effectuées précédemment. Dans un premier, il a fallu réduire la zone dédié au graphique. Pour cela il a fallu réduire les attributs permettant de définir les dimensions de la zone, mais il a aussi fallu redéfinir les dimensions des échelles permettant la construction des axes et le placement des points. Si les échelles n'avaient pas été redéfinies, les zones dédiées aux graphiques étaient automatiquement réajuster selon les éléments qu'elles possédaient. Une fois cette étape effectué, étant donné que les zones dédiés au graphiques furent plus petites, il a été nécessaire de repositionner correctement ces zone dans leur cellule en effectuant de léger décalage horizontal et vertical dans l'attribut transform. La dernière modification effectuée fut de réduire la taille

de police des valeurs des axes. Une fois ces étapes faites, on obtient le résultat souhaité c'est-à-dire un nuage de points dans chaque cellule.



Interactivité

Une des interactivités demandait au cours de ce projet, fut d'avoir la possibilité d'exporter les graphiques en format HTML tout en gardant les fonctionnalités d'interactivité précédente. La méthode utilisé fut d'utiliser d'exporter ces graphies via un URLs de données, c'est-à-dire de transmettre les différents éléments nécessaire à l'affichage au sein même de l'URL. Afin de pouvoir effectuer cette exportation, un bouton a été ajouté à la page HTML, ce bouton permet d'activer la fonction s'occupant de l'exportation. La fonction d'exportation crée alors une nouvelle balise quelconque qui servira à stocker l'URL contenant les informations à exporter. Une fois cette balise créé, on stocke la zone dédié au SVG grâce à son ID, défini lors de sa création, dans une variable. Les fonctions permettant l'interactivité sont également stockés dans une autre variable afin de pourvoir garder l'interactivité du graphique. Le CSS et la zone permettant l'affichage des points sont aussi récupéré et stocker. Une fois ces différents éléments stockés, elles sont combinés afin de ne faire qu'une seule et unique chaine de caractère qui servira à l'exportation. Ensuite, on définit alors que la nouvelle balise créé au début de la fonction aura comme fonctionnalité d'effectuer un téléchargement et on lui fournit le nom du fichier à télécharger. L'étape suivante consiste à définir la référence de la balise, pour cela on construit un URL en utilisant la méthode « data:text/html », qui permet de définir que le document sera un document HTML, et en ajoutant la chaine de caractère produite précédemment. On simule alors un clic utilisateur sur la balise afin de lancer le téléchargement du fichier. On obtient alors l'affichage des graphiques dans un fichier HTML a part ne nécessitant plus l'utilisation du logiciel R et ayant une taille de fichier largement inférieur à celui d'origine.



Une fois le téléchargement effectué on supprime la balise afin de ne pas garder d'élément superficiel dans la page HTLM de départ.

Pourtant cette fonctionnalité ne fonctionna pas sur certains navigateurs internet (exemple : Firefox). En effet les normes sur l'encodage de ces URLS de données n'étant pas identiques entre les différents navigateurs cela engendré un refus de compréhension du fichier chez certains navigateurs. Il a alors fallu ajouter une étape à l'exportation qui a eu pour but d'encoder la chaine de caractère servant de données pour l'URL. Pour cela la chaine de caractère est d'abord fournit à la fonction encodeURIComponent() qui permet de remplacer chaque exemplaire de certains caractères par des séquences d'octets correspondant. Cette fonction renvoie alors une nouvelle chaine de caractère encodé qui servira alors à la création de l'URL.

```
function download(){
	var a = document.body.appendChild(
		document.createElement("a")
	);
	a.setAttribute("display","none");
	var css ="<style type=\"text\/css\"> " +
document.getElementById("css").innerHTML + "</style>";
	var header="<header><h4 id=\"grid-ref\"></h4></header>";
	var fonct= "<script>" + document.getElementById("funct").innerHTML +
"<\/script>";
	var svg=document.getElementById("svggraph").outerHTML;
	var fichier=encodeURIComponent(css+header+fonct+svg);
	a.download = "export.html";
	a.href = "data:text/html," + fichier ;
	a.click();
	a.parentNode.removeChild(a);
}
```