Visualisation de données grâce à d3.js

Structure du programme:

Avant de se concentrer sur les résultats, nous allons nous intéresser à la structure du programme en elle-même. Le programme commence par la création d'une application qui cherchera à renvoyer svgData dans localhost:3000
svgData va être notre visualisation et sera définie plus loin.

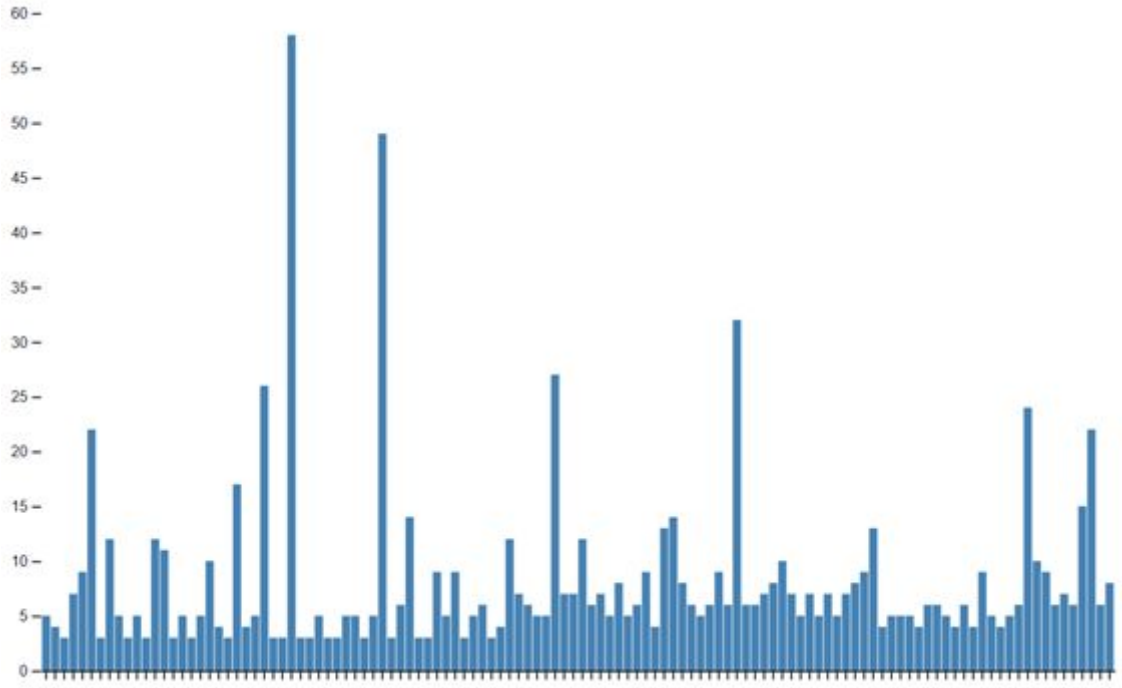
```
app.get('/', (req, res) => {
    res.send(svgData)
})
```

On a ensuite un code qui lie les fichiers nécessaires, organise les données comme cela nous arrange, puis fait appel aux fonctions getMapData et getMap qui permettent respectivement de récupérer les données nécessaires pour les visualisations et de donner à svgData la valeur nécessaire pour être affiché dans l'application. Après avoir donné à svgData la valeur souhaitée, on fait appel à l'application qui va nous permettre d'afficher nos diagrammes.

```
fs.readFile("TP3.csv", 'utf8', function (err, csvFile) {
    fs.readFile("states.json", 'utf8', function (err2, jsonFile) {
        if (err || err2) {
            return (err) ? console.log(err) : console.log(err2)
        }

        let dsvParser = d3.dsvFormat(';')
        let csvData = dsvParser.parse(csvFile, d3.autoType)

        let jsonData = JSON.parse(jsonFile)


        const dom = new JSDOM(`<!DOCTYPE html><body></body>`);
        let body = d3.select(dom.window.document.querySelector("body"))

        let csvDataMap1 = getMap1Data(csvData)
        let csvDataMap2 = getMap2Data(csvData)
        svgData = getMap1(csvDataMap1, jsonData, body)
        //svgData = getMap2(csvDataMap2, body)

        app.listen(port, () => {

            console.log(`Example app listening at http://localhost:${port}`)
        })
    })
})
```

Première tentative de visualisation:

Ayant eu du mal à comprendre le cours, nous avons décidé de commencer par faire une première visualisation simple. Nous sommes allés sur le site https://observablehq.com/@d3/gallery pour trouver des visualisations intéressantes et relativement simples. Après avoir compris le code sur le site, nous l'avons copié collé et adapté aux données dont nous disposons. Cette première visualisation représente le nombre de morts dans chaque tuerie et les répartis sur un bar chart.



Une deuxième visualisation plus poussée:

Nous sommes ensuite retournés sur le site web avec différentes représentations et nous avons voulu représenter toutes ces tueries sur une carte. Grâce à notre fichier "states.json" nous traçons une carte des Etats-Unis puis les contours des différents états. Pour chacun des tueries, on récupères ses coordonnées et son nombre de victimes grâce à cette fonction:

```
function getMap1Data(csvData) {
    return csvData.map(d => { return { victims: d.total_victims, latitude: d.latitude, longitude: d.longitude } })
}
```

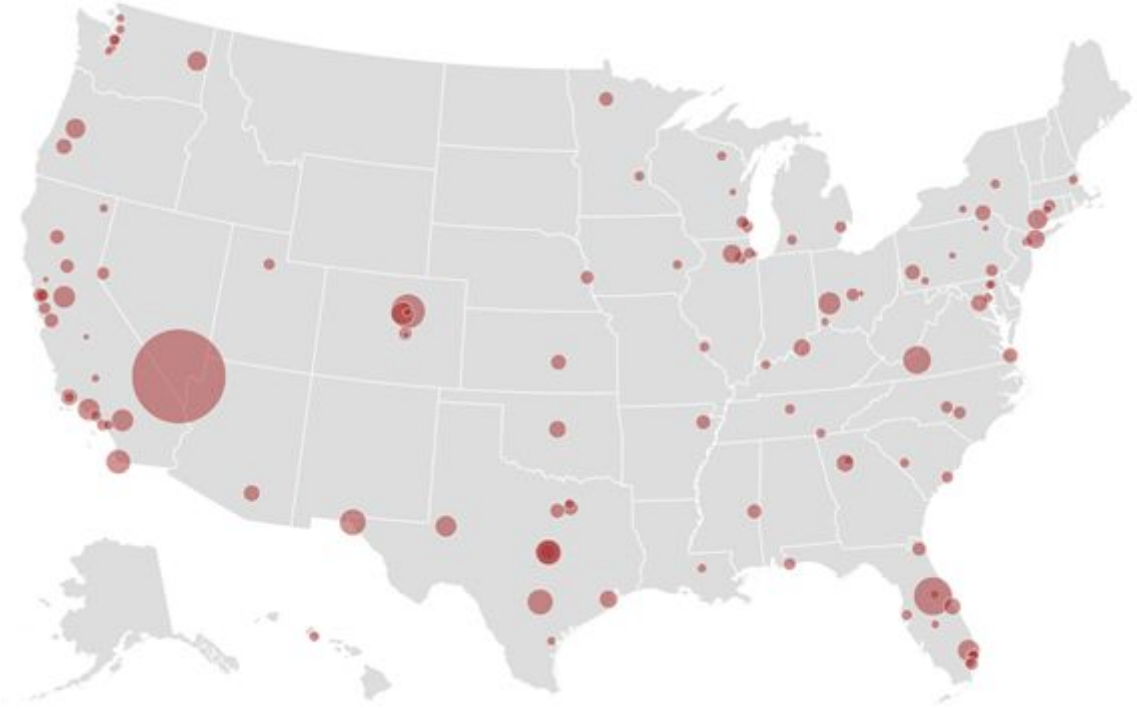
On adapte ensuite la taille des cercles en fonction du nombre de victimes avec cette fonction:

```
let radius = d3.scaleSqrt([0, d3.max(data, d => d.victims)], [0, 40])
```

Pour finir, on adapte les coordonnées des tueries à la taille de la carte avec cette fonction:

```
let projection = d3.geoAlbersUsa().scale(1300).translate([487.5, 305])
```

On obtient alors cette carte comme résultat final:

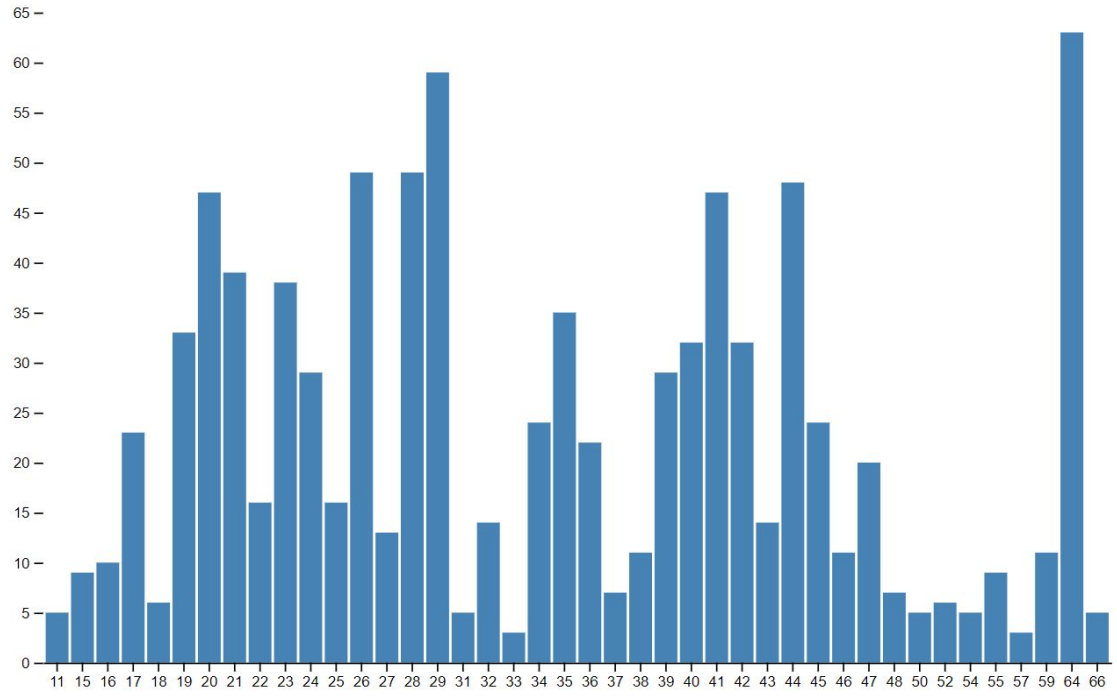


Améliorations post-présentation:

Ces deux représentations sont celles que nous avons montrées lors de la présentation orale. Après cette présentation, nous avons tenu à revenir sur certains points de notre présentation à améliorer. Nous sommes tout d'abord revenus sur notre premier graphique, nous voulions qu'il n'ait pas autant de barres et qu'il y ait des labels pour chaque barre. Nous avons alors voulu représenter le nombre de victimes regroupées par l'âge du tueur. Pour cela, nous avons créé une fonction "getMap2Age" qui va parcourir nos données et crée une nouvelle liste avec les données qui nous arrangent. On trie ensuite ces données par rapport à l'âge grâce à cette petite fonction:

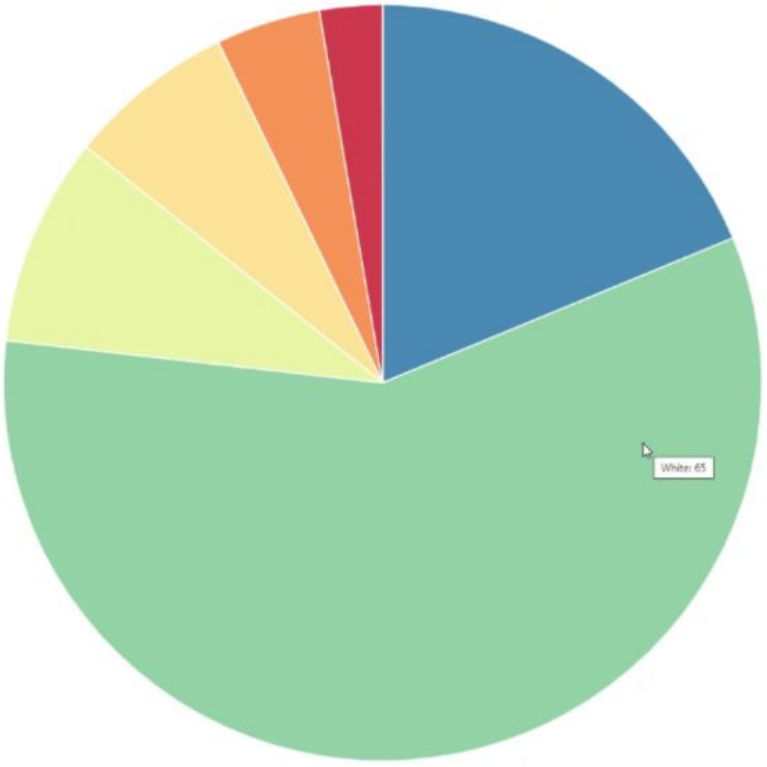
```
csvDataMap2Age.sort(function(first, second) {
    return first.age - second.age;
});
svgData = getMap2(csvDataMap2Age.slice(2), body)
```

On obtient alors ce graphique:



Un graphique supplémentaire:

Nous avons voulu faire un diagramme qui montre le nombre de tueries par race. De la même manière que nous avons regroupé les meurtriers par âge à la visualisation précédente, nous regroupons ici les tueurs par race. On obtient alors ce graphique, on peut voir à quoi correspond chaque partie en laissant la souris sur le graphique.



Comment se servir de notre programme?

Notre programme n'affiche qu'une visualisation à la fois, pour afficher une visualisation plutôt qu'un autre, il faut aller vers les lignes 50, on trouvera ceci.

```
//svgData = getMap1(csvDataMap1, jsonData, body)

/*
let csvDataMap2Age = getMap2Age(csvDataMap2)
csvDataMap2Age.sort(function(first, second) {
    return first.age - second.age;
});
svgData = getMap2(csvDataMap2Age.slice(2), body)
*/

let csvDataMap3Race = getMap3Race(csvDataMap3)
console.log(csvDataMap3Race)
svgData = getMap3(csvDataMap3Race, body)
```

Il faut mettre en commentaire les parties que l'on ne veut pas voir, getMap1 permet d'afficher la carte, getMap2 et les quelques lignes qui suivent permettent d'afficher le bar chart, et les quelques dernières lignes permettent d'afficher le pie chart. Sur cette capture, le programme cherchera à afficher le pie chart en localhost 3000.

Conclusion:

Ce projet à été relativement dur car nous avons eu du mal à suivre les cours. Nous avons dû prendre beaucoup des notions du cours depuis 0 ce qui à fait que nous avons pris beaucoup de temps pour faire chacune des parties. Cependant ce projet a rempli ses objectifs car il a été instructif et nous avons réussi à produire des résultats.