Cours: VISUALISATION DE DONNEES

Master Informatique Visuelle
2022/2023

Prof. Slimane Larabi
USTHB

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction

2.1 Librairie D3js (Data Driven Documents)

2.1.1 Introduction à D3js

2.1.2 Visualisation de flux de données

2.1.3 Design et Interaction guidés par les données

2.2 Librairies de Python

2.3 Projet

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction

Airline on-time performance

Exemple de jeu de données disponible

Les données se composent des détails d'arrivée et de départ de tous les vols commerciaux aux États-Unis, d'octobre 1987 à avril 2008. Il s'agit d'un grand ensemble de données : il y a près de 120 millions d'enregistrements au total et occupe 1,6 gigaoctets d'espace compressé et 12 gigaoctets. lorsqu'il n'est pas compressé.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction

Le challenge

L'objectif de l'exploitation de données est de fournir un résumé graphique des caractéristiques importantes de l'ensemble de données.

Ce qu'on peut essayer de retrouver sur la visualisation:
- Quel est le meilleur moment de la journée/jour de la semaine/période de l'année pour voler afin de minimiser les retards ?

- Les avions plus anciens subissent-ils plus de retards ?

- Comment le nombre de personnes voyageant entre différents endroits change-t-il au fil du temps ?

- Dans quelle mesure la météo prédit-elle les retards d'avion ?

- Peut-on détecter des défaillances en cascade lorsque des retards dans un aéroport entraînent des retards dans d'autres ? Y a-t-il des liens critiques dans le système ?

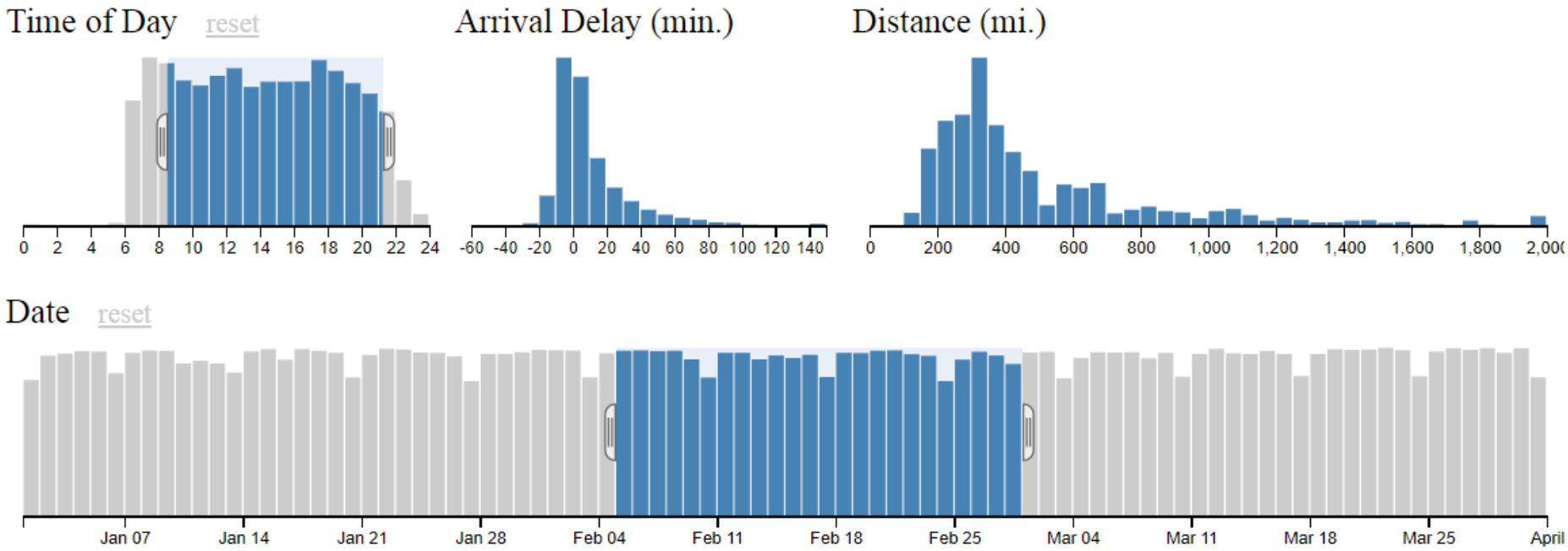
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction

On peut travailler avec des sous-ensembles:
comparer les patterns de vol avant et après le 11 septembre,
ou entre la paire de villes entre lesquelles on voyage le plus souvent,
ou tous les vols à destination et en provenance d'un aéroport majeur comme Chicago (ORD).

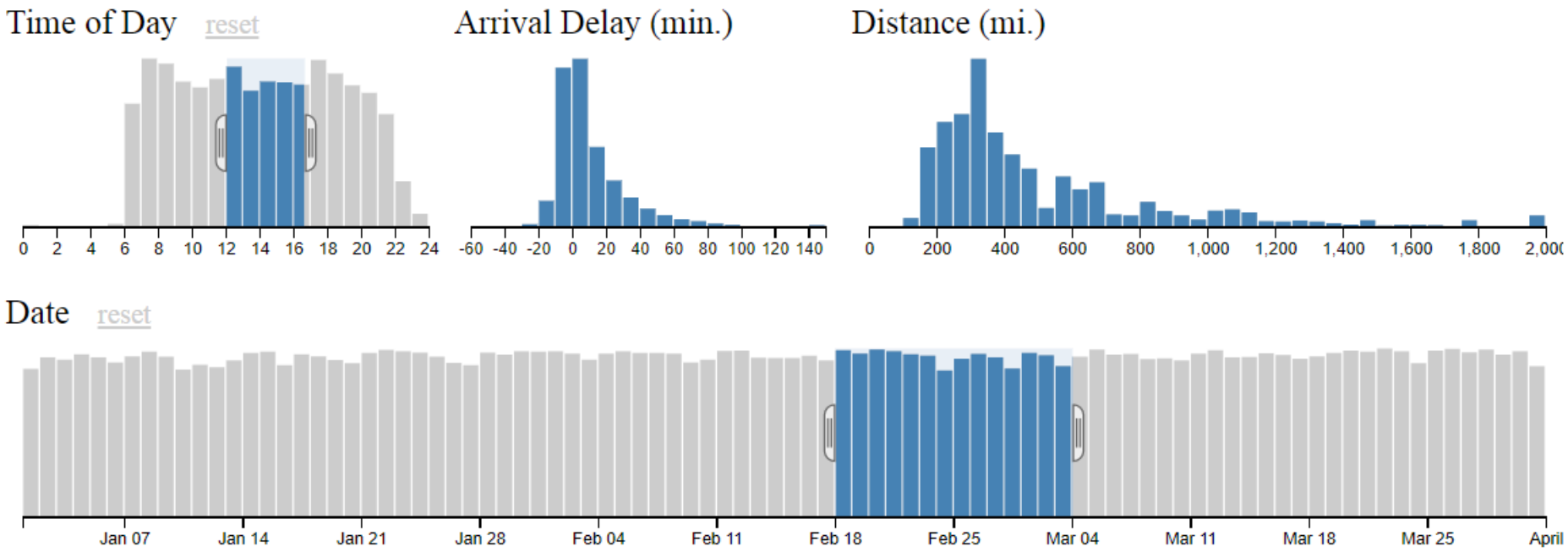
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction



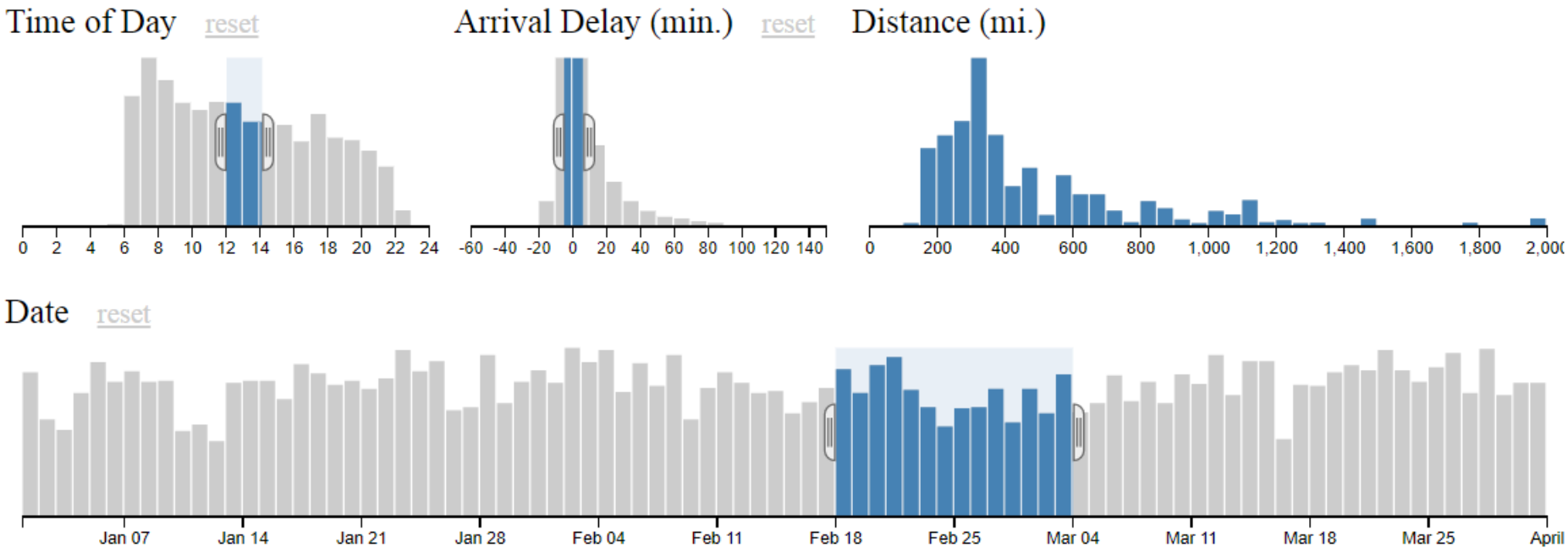
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.0 Introduction



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1 Librairie D3js
2.1.1 Introduction à D3js

D3 qui fait référence à Data-Driven Documents, ou D3.js est une librairie pour la création de visualisation de données sur le web.

D3 facilite la génération et la manipulation de documents web:
- Charger les données dans la mémoire du navigateur
- Lier les données à des éléments à l'intérieur du document

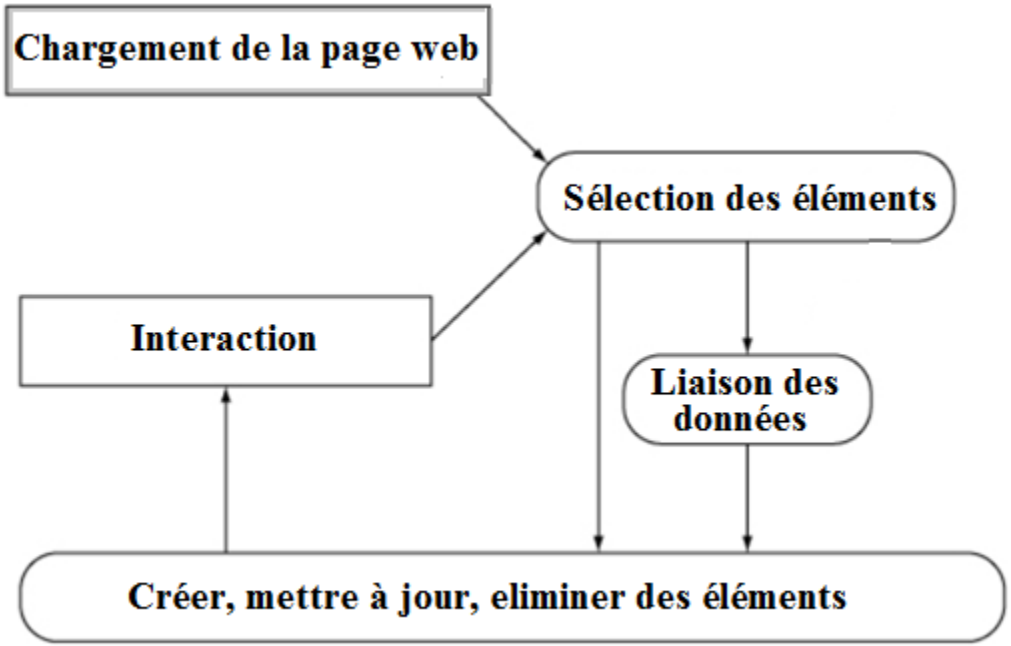
Elle a été développé par **Mike Bostock**, d'autres contributeurs ont rejoint et la librairie est actuellement open source sous licence BSD.

Ci-après les principales caractéristiques de D3js.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **D3 infère l'apparence des pages web à partir de la liaison de données**



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

Les éléments de page web peuvent être modifiés

L'élément de base qui représente un rectangle dans lequel on peut insérer un paragraphe, liste, table

Il peut être sélectionné et modifié (comme un pays dans une carte, ou cercle et ligne dans une visualisation).

d3.select() c'est la syntaxe à utiliser pour la mise à jour (création et mouvement de cercles ou).

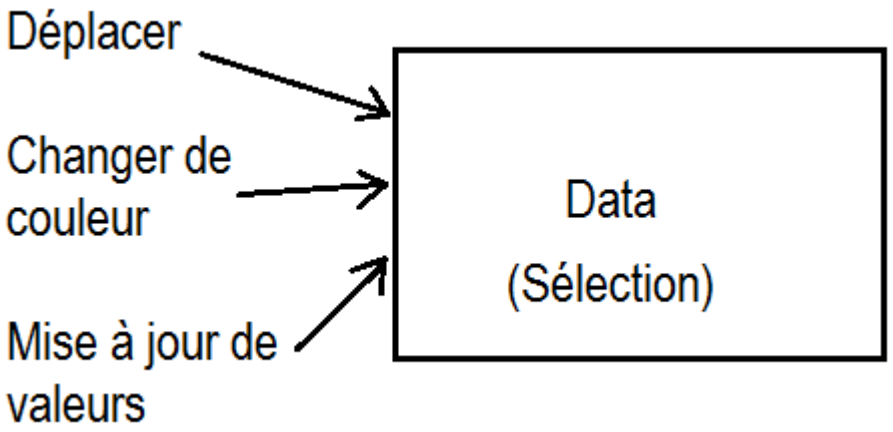
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **D3 permet la sélection et la liaison de données**

Etant donné un ensemble de données et un ensemble d'éléments constituant une page web (graphique, (élément de division du contenu) , <p>),

On peut les représenter avec du texte ou tailles, ou des couleurs sur le document web.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

d3.**select**(“element_to_select).

d3.select()

Cette fonction dans D3.js est utilisée pour sélectionner le premier élément qui correspond à la chaîne du sélecteur spécifiée.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

```
<!DOCTYPE html> [1]
<html>
<head><script src="https://d3js.org/d3.v4.min.js"></script>
</head>
<body><p>GeeksforGeeks</p>
<p>A computer science portal for geeks</p>

<script>// Calling the select() function
        var a = d3.select("p").text();
        console.log(a); //Output: “GeeksforGeeks”
</script>
</body></html>
```

[1] https://www.geeksforgeeks.org/d3-js-d3-select-function/

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

d3.**selectAll**("element_to_select).

The **d3.selectAll() function** in D3.js is used to select all the element that matches the specified selector string.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

```
<!DOCTYPE html> [1]
<html>
<head>
        <script src = "https://d3js.org/d3.v4.min.js"></script>
</head>
<body>  <div>Geeks</div>
        <div>GeeksforGeeks</div>
        <script>
                // Calling the selectAll() function
                d3.selectAll("div").text();
        </script></body></html>
```

Output on the web page

Geeks
GeeksforGeeks

Output:

Geeks
GeeksforGeeks

[1] https://www.geeksforgeeks.org/d3-js-d3-selectall-function/

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

d3.**selectAll**("circle.a").style("fill", "red").attr("cx", 100);
Pour tous les cercles de classe a, les translater avec dx=100 et les colorier en rouge.

d3.**selectAll**("div").style("background", "red").attr("class", "b");
Associer la couleur de fond (rouge) et la classe b à tous les éléments div.

d3.**selectAll**("div.market").data([1,5,11,3])
Associer les données aux div de classe market.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

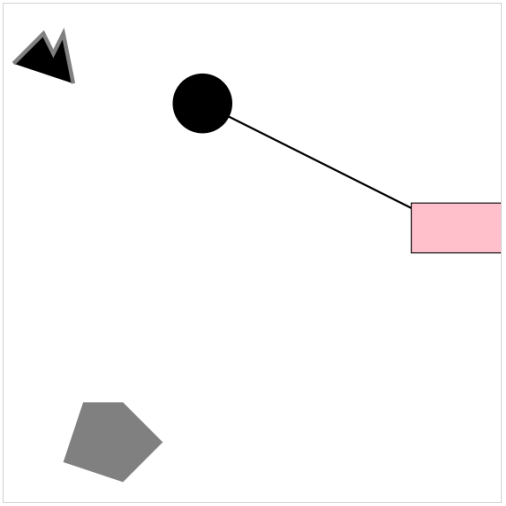
- **D3 simplifie l'intégration des graphiques SVG**

D3js permet le dessin SVG avec un certain niveau d'abstraction.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

```
<body>
 <div id="infovizDiv">
 <svg style="width:500px; height:500px; border: 1px lightgray solid;">
  <path d="M 10,60 40,30 50,50 60,30 70,80“ style="fill: black;
    stroke:gray;stroke-width:4px;" />
  <polygon style="fill: gray;"
    points="80,400 120,400 160,440 120,480 60,460" />
 <g>
 <line x1="200" y1="100" x2="450" y2="225"
 style="stroke: black; stroke-width:2px;"/>
 <circle cy="100" cx="200" r="30"/>
 <rect x="410" y="200" width="100" height="50"
    style="fill: pink; stroke: black; stroke-width:1px;" />
 </g>
 </svg>
</body>
```

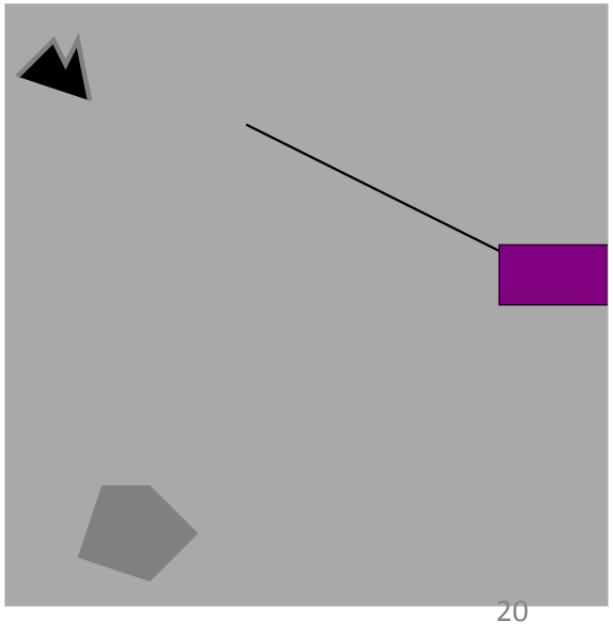


Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

Possibilité de manipuler les éléments SVG en utilisant le sélecteur JavaScript telque: **document.getElementById** ou bien D3, en l'éliminant, lui changeant de style.

```
d3.select("svg").style("background", "darkgray");
        d3.select("circle").remove()
        d3.select("rect").style("fill", "purple")
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- Method Chaining in D3

```
var bodyElement = d3.select("body");
var paragraph = bodyElement.append("p");
paragraph.text("Hello World!");
```

Ce code est équivalent au suivant:

```
d3.select("body").append("p").text("Hello World!");
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **Data Binding in D3**

data() : Joins data to the selected elements
enter() : Creates a selection with placeholder references for missing elements
exit(): Removes nodes and adds them to the exit selection.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **Data Binding in D3**

data() : Joins data to the selected elements

```
<body>
<p>D3 Tutorials 1</p>
 <p>D3 Tutorials2</p>
<script>
        var myData = [1, 2];
            var p = d3.select("body")
            .selectAll("p")
            .data(myData)
            .text(function (d) {return d; });
</script>
</body>
```

Output

1

2

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **Data Binding in D3**

.enter() est une fonction utilisée pour définir ce qu'il faut faire avec chaque élément supplémentaire du tableau de données.

```
<body>
<script>
   var data = [4, 1, 6, 2, 8, 9];
   var body = d3.select("body")
            .selectAll("span")
//<span> tag is much like the <div> element, but <div> is a block-level element
and <span> is an inline element.
            .data(data)
            .enter()
            .append("span")
            .text(function(d) { return d + " "; });
</script>
</body>
```

Output: 416289

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **Data Binding in D3**

.exit()

exit is used to remove a node.

```
<body>
<p>par1</p> <p>par2</p> <p>par3</p>
<script>
var myData = ["Hello World!"];
var p = d3.select("body")
.selectAll("p")
.data(myData)
.text(function (d, i) { return d; })
.exit() .remove();
</script> </body>
```

```
<html lang="en">
▶ <head>…</head>
▼ <body> == $0
    <p>Hello World!</p>
  ▶ <script>…</script>
  </body>
</html>
```

.exit().remove()

remove additional <p> elements because data array includes only one data value.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

- **D3 opère sur les éléments de Tableaux moyennant les Fonctions**

```
someNumbers = [17, 82, 9, 500, 40];
someColors = ["blue", "red", "chartreuse", "orange"];

somePeople = [{name: "Peter", age: 27}, {name: "Sulayman", age: 24},
{name: "K.C.", age: 49}];

someNumbers.filter (function(el) {return el >= 40});
```

La fonction .filter() est une méthode du tableau et accepte une fonction qui itère sur le tableau par la variable portée.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

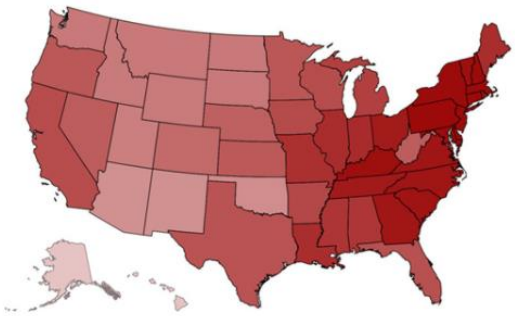
D3 opère sur différents types de données

- **Données brutes**
- **Données tabulées**
  - d3.csv(), comma-delimited files
  - d3.tsv(), tab-delimited files
  - d3.dsv(). allows you to declare the delimiter
- **Données hiérarchisées (nested)**
- **Données de graphes,**
- **Données des réseaux**
- **Données géographiques**

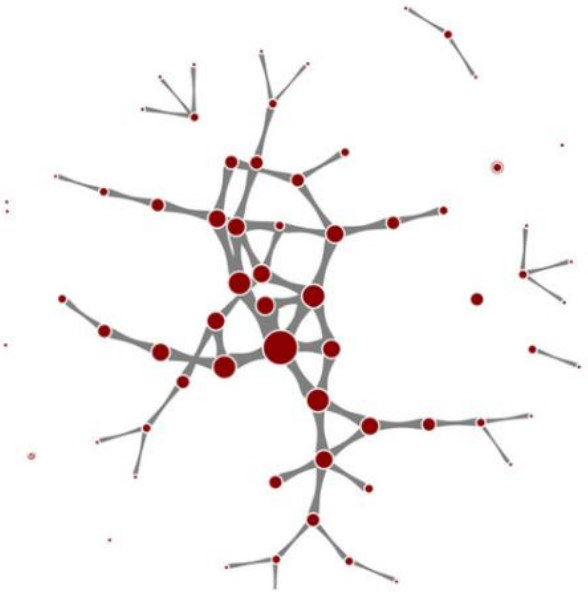
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.1 Introduction à D3js

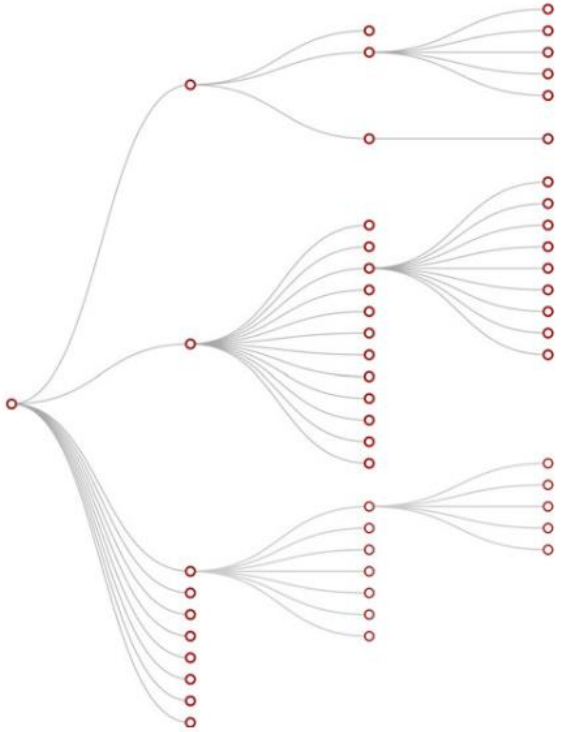
Geographic data



Network data



Nested data

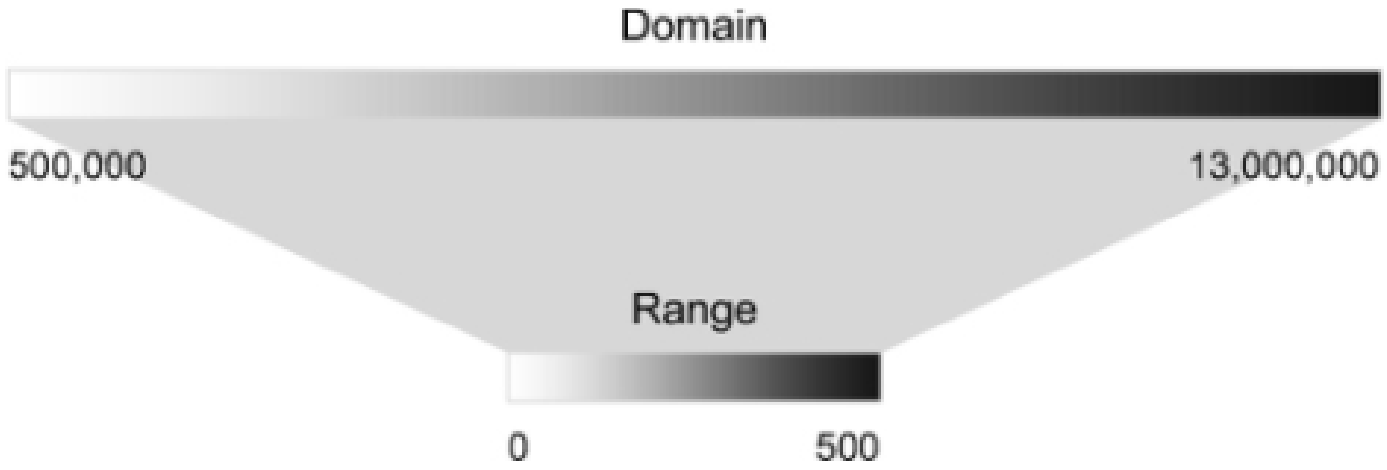


Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

b. Transformation de données

Scales and scaling: Changement d'échelle



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

b. Transformation de données

Scales and scaling

Les échelles dans D3 mappent un ensemble de valeurs (le domaine) à un autre ensemble de valeurs (la plage) dans une relation déterminée par le type d'échelle choisi.

```
var newRamp = d3.scaleLinear().domain([5000,13000]).range([0, 50]);
        console.log(newRamp(6000));
        console.log(newRamp(9000));
        console.log(newRamp.invert(31.3));
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

b. Transformation de données

Scales and scaling

Les échelles dans D3 mappent un ensemble de valeurs (le domaine) à un autre ensemble de valeurs (la plage) dans une relation déterminée par le type d'échelle choisi.

```
var newRamp = d3.scaleLinear().domain([5000,13000]).range([0, 50]);
        console.log(newRamp(6000));
        console.log(newRamp(9000));
        console.log(newRamp.invert(31.3));
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

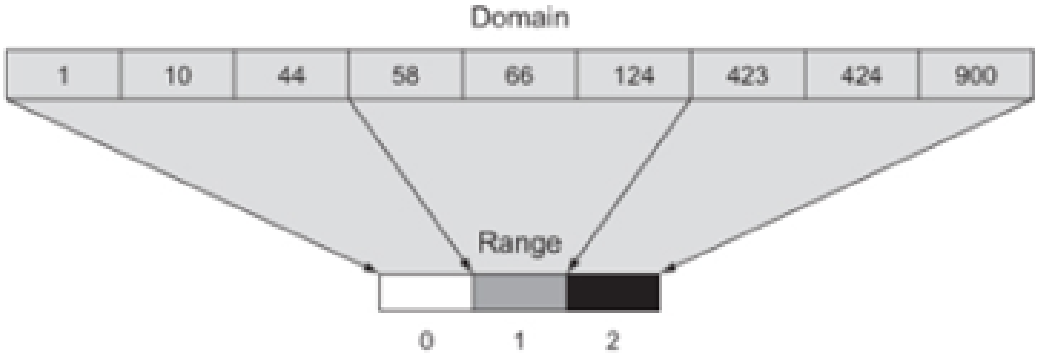
2.1.2 Visualisation de flux de données

b. Transformation de données

Binning: catégorisation des données

```
var sampleArray = [423,124,66,424,58,10,900,44,1];
var qScale = d3.scaleQuantile().domain(sampleArray).range([0,1,2]);
qScale(423);
qScale(20);
qScale(10000);
```

Quantile scales prennent une plage de valeurs et leur réaffectent un ensemble de plages de taille égale



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

c. Mesure des données

```
d3.csv("cities.csv", data => {
        d3.min(data, el => +el.population);
        // d3.min(data, function(d) { return d; })
        d3.max(data, el => +el.population);
        d3.mean(data, el => +el.population);
        d3.extent(data, el => +el.population); // return [min, max]
});
```

```
500000 13000000 6856250
▼(2) [500000, 13000000] i
    0: 500000
    1: 13000000
    length: 2
  ▶ [[Prototype]]: Array(0)
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

c. Mesure des données

```
var testArray =  [88, 10000, 1, 75, 12, 35];
        d3.min(testArray, el => el);
        d3.max(testArray, el => el);
        d3.mean(testArray, el => el);
```

```
1 10000 1701.8333333333333
>
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Selections and binding

```
d3.csv("cities.csv", (error,data) => {
  if (error) {   console.error(error)   }
  else {   dataViz(data)   }
  });
function dataViz(incomingData) {
 d3.select("body").selectAll("div.cities")
   .data(incomingData)
   .enter()
   .append("div")
   .attr("class","cities")
   .html(d => d.label);
}
```

San Francisco
Fresno
Lahore
Karachi
Rome
Naples
Rio
Sao Paolo

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Selections and binding

```
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("p")
.data(dataset)
.enter()
.append("p")
.text(function(d) { return ("Le paragraphe numéro"+d);});
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Selections and binding

Changement dynamique de style de paragraphes

```
.text(function(d) { return "Le paragraphe  " + d + " en
couleur rouge"; })
.style("color", "red");
```

Le paragraphe 5 en couleur rouge

Le paragraphe 10 en couleur rouge

Le paragraphe 15 en couleur rouge

Le paragraphe 20 en couleur rouge

Le paragraphe 25 en couleur rouge

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Selections and binding

Changement dynamique de style de paragraphes

```
.text(function(d) { return "La couleur du paragraphe est rouge si la
valeur suivante" + d + "est > 15"; })

.style("color", function(d) {
if (d > 15) { //Threshold of 15
                return "red";
                }
else { return "black"; }
});
```

La couleur du paragraphe est rouge si la valeur suivante5est > 15

La couleur du paragraphe est rouge si la valeur suivante10est > 15

La couleur du paragraphe est rouge si la valeur suivante15est > 15

La couleur du paragraphe est rouge si la valeur suivante20est > 15

La couleur du paragraphe est rouge si la valeur suivante25est > 15

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Graphe (Histogrammes) à partir des données

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>D3: Creating paragraphs dynamically from data</title>
        <script type="text/javascript" src="d3.v3.js"></script>
        <style type="text/css">

            div.barre {
                display: inline-block;
                width: 40px;
                height: 175px; /* Hauteur sera changée */
                background-color: red;
            }

        </style>
    </head>
    <body>
</html>
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Graphe (Histogrammes) à partir des données

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <body>
    <script type="text/javascript">

            var dataset = [ 5, 10, 45, 20, 125, 10, 45 ];


            d3.select("body").selectAll("div")
                .data(dataset)
                .enter()
                .append("div")
                .attr("class", "barre")
                .style("height", function(d) {return d + "px";});

        </script>
    </body>
</html>
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

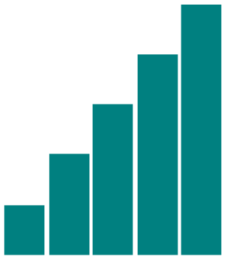
Graphe (Histogrammes) à partir des données

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Graphe (Histogrammes) à partir des données



```
<style type="text/css">
div.bar {display: inline-block;
        width: 20px;
        height: 75px;
        background-color: teal;
        margin-right: 2px;} </style>
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("div")
.data(dataset).enter().append("div").attr("class", "bar")
.style("height", function(d) { var barHeight = d * 5;
                               return barHeight + "px";  });
```

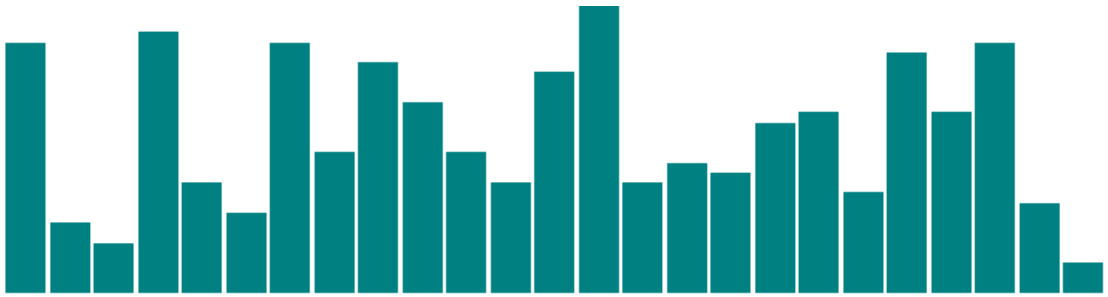
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Graphe (Histogrammes) à partir des données

var dataset = [25, 7, 5, 26, 11, 8, 25, 14, 23, 19, 14, 11, 22, 29, 11, 13, 12, 17, 18, 10, 24, 18, 25, 9, 3];
Avec le même code précédent, on obtient:



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Avec SVG: Graphe (Histogrammes) à partir des données

```
var w = 500;
var h = 100;
var dataset = [ 5, 10, 13, 19, 61, 25, 22, 18, 35, 13, 11, 12, 45, 20, 18, 17, 16, 78, 23, 25 ];
//Create SVG element
var svg = d3.select("body")
.append("svg")
.attr("width", w)
.attr("height", h);
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Avec SVG: Graphe (Histogrammes) à partir des données

```
svg.selectAll("rect")
        .data(dataset)
        .enter()
        .append("rect")
        .attr("x", function(d,i){return i*21;})
        .attr("y", function(d,i){return h-d;})
        .attr("width", 20)
        .attr("height", function(d,i){return d;});
```

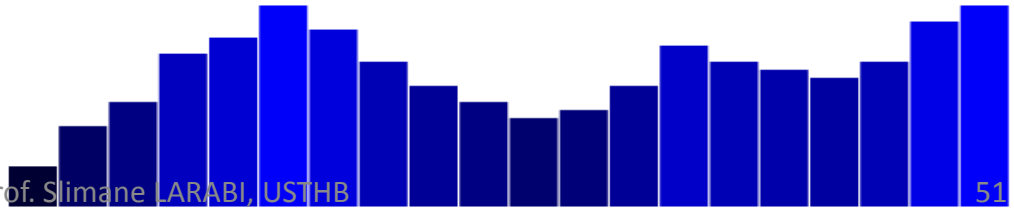
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Ajout de style:

```
svg.selectAll("rect")
            .data(dataset)
            .enter()
            .append("rect")
            .attr("x", function(d,i) {return i*21;})
            .attr("y", function(d,i) {return h-d;})
            .attr("width", 20)
            .attr("height", function(d,i) {return d;})
.attr("fill", function(d) {return "rgb(0, 0, " + (d * 10) + ")";});
```



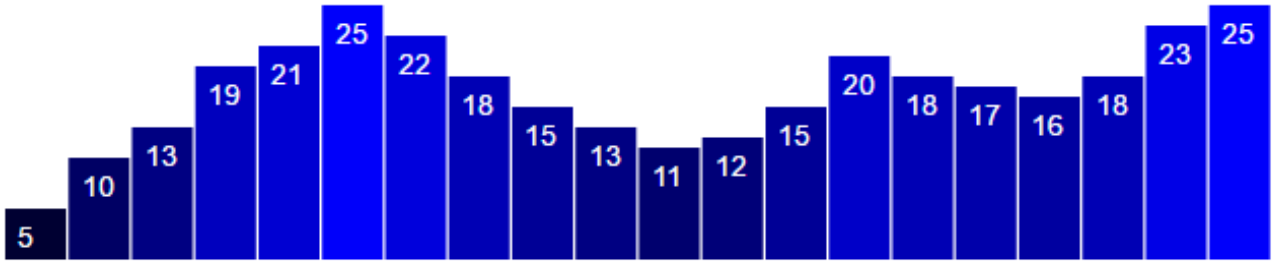
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.2 Visualisation de flux de données

d. Liaison de données

Ajout de text:

```
svg.selectAll("text")
.data(dataset)
.enter()
.append("text")
.text(function(d) {return d;})
        .attr("x", function(d, i) {
        return i * (w / dataset.length);})
        .attr("y", function(d) {return h - (d * 4); });
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Interaction via évènements

```
<p>Click on this text to update the chart with new data values
(once).</p>

d3.select("p")
        .on("click", function() {
//  Lecture des données, changement d'échelle, visualisation des
rectangle…
                });
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Transitions

```
// ici on a fait la visualisation de dataset
//mise à jour des rectangles, suite à nouvelles données
svg.selectAll("rect")
.data(dataset)
.transition()

// sans cette indication (.transition),
// la visualisation se fera d'une façon brusque!

.attr("y", function(d) { return h - yScale(d); })
.attr("height", function(d) {return yScale(d);})
.attr("fill", function(d) {return "rgb(0, 0, " + (d * 10) + ")";});
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Duration

duration() est specifiée juste après ".transition()",

Elle spécifie en milliseconds la durée de la transition.
1 seconde=1000 ms.

En fonction de la valeur indiquée, on peut avoir une longue transition ou brève.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Duration

Pour faire suivre le texte dans la transition, on utilise la même durée.

```
svg.selectAll("text")
.data(dataset)
.transition()
.duration(5000)
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Nature du mouvement : ease()

```
svg.selectAll("rect")
.data(dataset)
.transition()
.duration(2000)
.ease("linear")
```

On peut utliser plusieurs attributs:

"linear" "circle" "elastic" "bounce" "**cubic-in-out**“ “delay”

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

delay()

delay() specifies quand la transition commence.

```
.transition()
.delay(1000) //1,000 ms or 1 second
.duration(2000) //2,000 ms or 2 seconds
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

delay()

```
.delay(function(d, i) {
return i * 100;})
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Début et fin de transition

```
//Update all circles
svg.selectAll("circle")
.data(dataset)
.transition()
.duration(1000)
.each("start", function() {
//.each method lets you call a function for each element of a selection.
        d3.select(this)
        .attr("fill", "magenta")
        .attr("r", 3);
        })
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

Début et fin de transition

```
.attr("cx", function(d) {
return xScale(d[0]);
})
.attr("cy", function(d) {
return yScale(d[1]);
})
.each("end", function() {
        d3.select(this)
        .attr("fill", "black")
        .attr("r", 2);
        });
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données

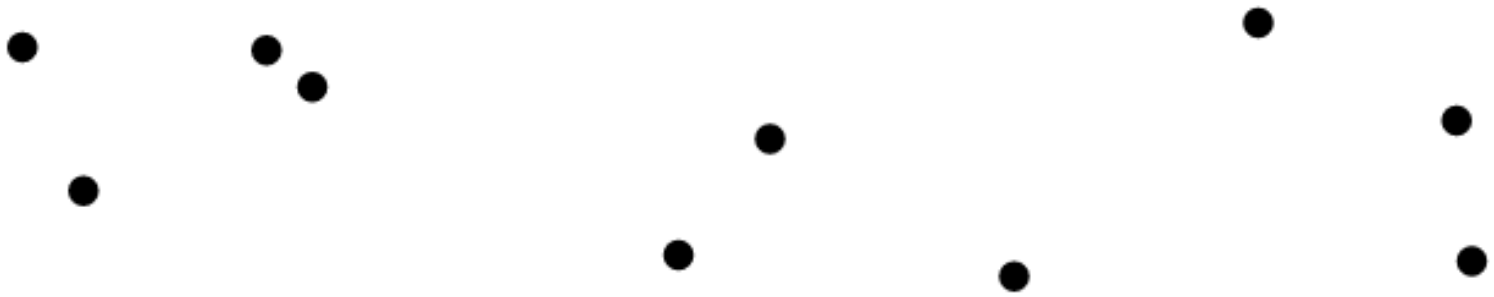
Ajout de diques dans l'élément SVG (cercles pleins)

```
svg.selectAll("circle")
        .data(dataset)
// dataset =[[4,8][32,66] ...]    d[0]=4, d[1]=8
        .enter()
        .append("circle")
        .attr("cx", function(d=d[0] et d[1]) {return d[0];})
        .attr("cy", function(d) {return d[1];})
        .attr("r", 5);
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données

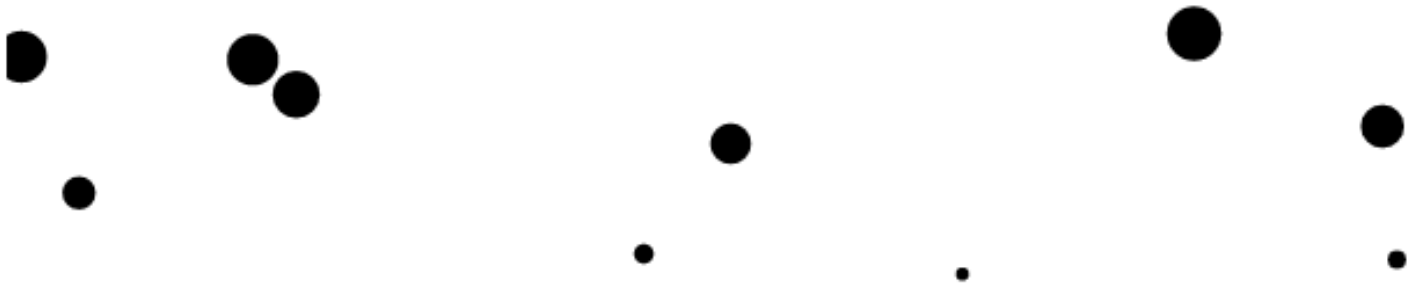
Rayons comme attributs des cercles

```
.attr("r", function(d) { return Math.sqrt(h - d[1]);   })
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données

Ajout aux cercles de labels (text)

```
svg.selectAll("text")
.data(dataset)
.enter()
.append("text")
.text(function(d) {return d[0] + "," + d[1];})
.attr("x", function(d) {return d[0]; })
  .attr("y", function(d) {return d[1];  })
 .attr("font-family", "sans-serif")
.attr("font-size", "11px")
 .attr("fill", "red");
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Eléments SVG à partir des données

Amélioration de la position du texte:
.attr("x", function(d) {return 3+(Math.sqrt(h - d[1]))/2+d[0]; })

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Changement d'échelles

Le changement d'échelle est nécessaire pour représenter les données:
passer d'un espace de données réelles vers un espace écran.

Input domain: l'intervalle des données en entrée.
Output range: L'intervalle des données à visualiser en utité pixels.

Normalisation consiste à obtenir un Output range de 0..1

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Changement d'échelles

```
var scale = d3.scale.linear();
scale.domain([100, 500]);     // Intervalle des données en entrée
scale.range([10, 250]); // Intervalle de visualisation
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Changement d'échelles

Définition d'échelles dynamiques:
On ramènera toutes les valeurs dans l'espace de SVG (w,h)

```
var xScale = d3.scale.linear()
                .domain([0, d3.max(dataset, function(d) { return d[0]; })])
                .range([0, w]);

var yScale = d3.scale.linear()
                 .domain([0, d3.max(dataset, function(d) { return d[1]; })])
                 .range([0, h]);
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Changement d'échelles

Définition d'échelles dynamiques:

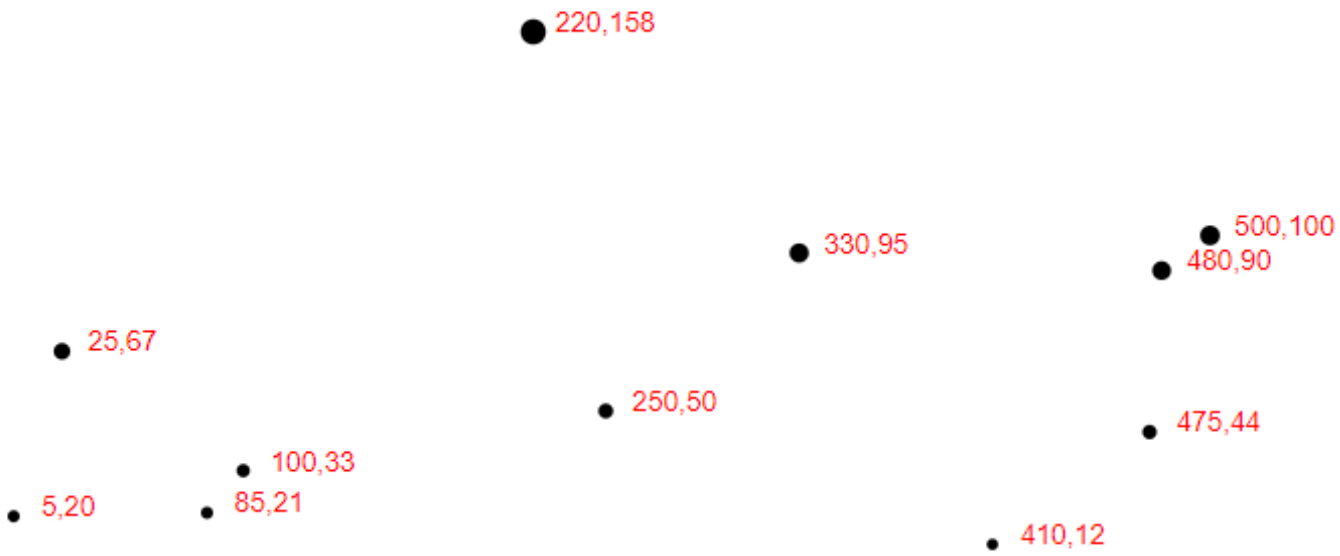
Pour faire référence à la valeur dans le nouveau domaine:

xScale(d[o]);

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Graphe avec Changement d'échelles



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Ajout des Axes à Graphe

Les fonctions de D3 génèrent les *axes incluant les* lignes, les labels, et les graduations.

Les axes générés par la fonctions supporte les valeurs quantitatives.

Au minimum, on doit spécifier à la fonction quelle échelle utiliser.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Ajout des Axes à Graphe

```
//Définir l'axe des X
var xAxis = d3.svg.axis()
                .scale(xScale)
                .orient("bottom")
                .ticks(5);
```

Les labels apparaissent par défaut en bas de l'axe.

Les orientation possibles sont top et bottom pour l'axe horizontal.

Pour l'axe vertical, left et right sont utilisés.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Ajout des Axes à Graphe

La génération de l'axe se fait par appel à la fonction:

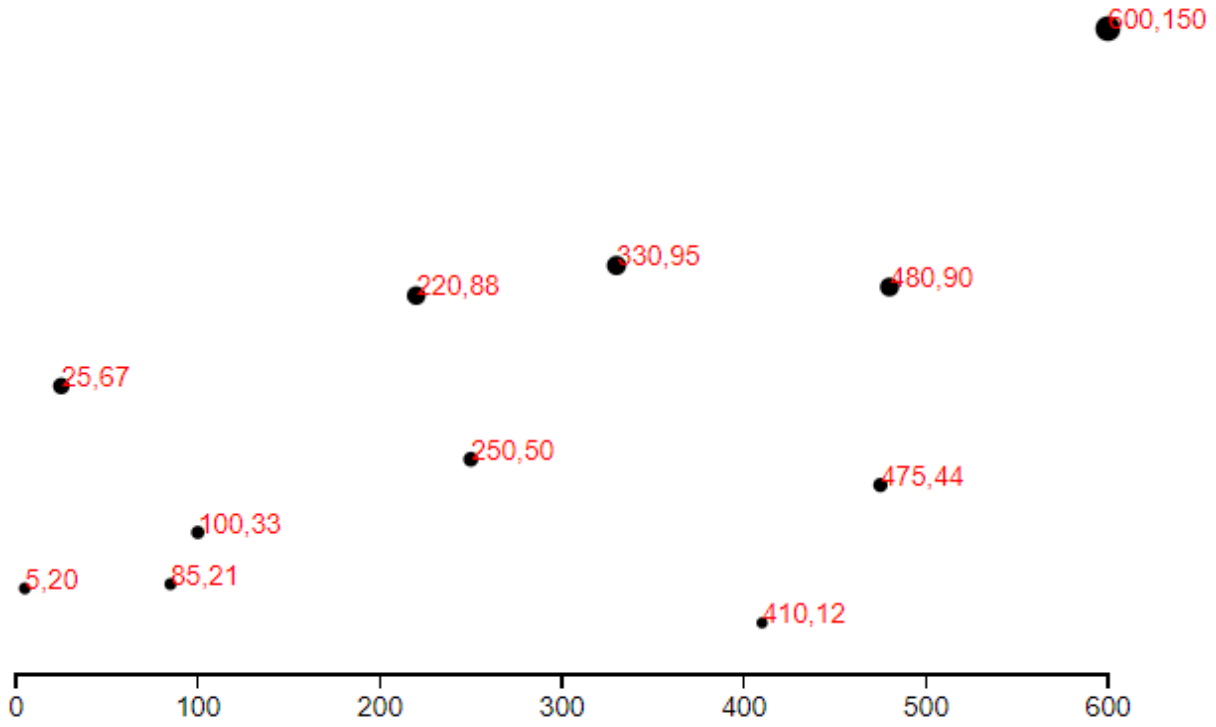
```
svg.append("g")
.attr("class", "axis")
.attr("transform", "translate(0," + (h - padding) + ")")
.call(xAxis);
```

Pour SVG, g signifie le *groupe d'é*léments.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Ajout des Axes à Graphe

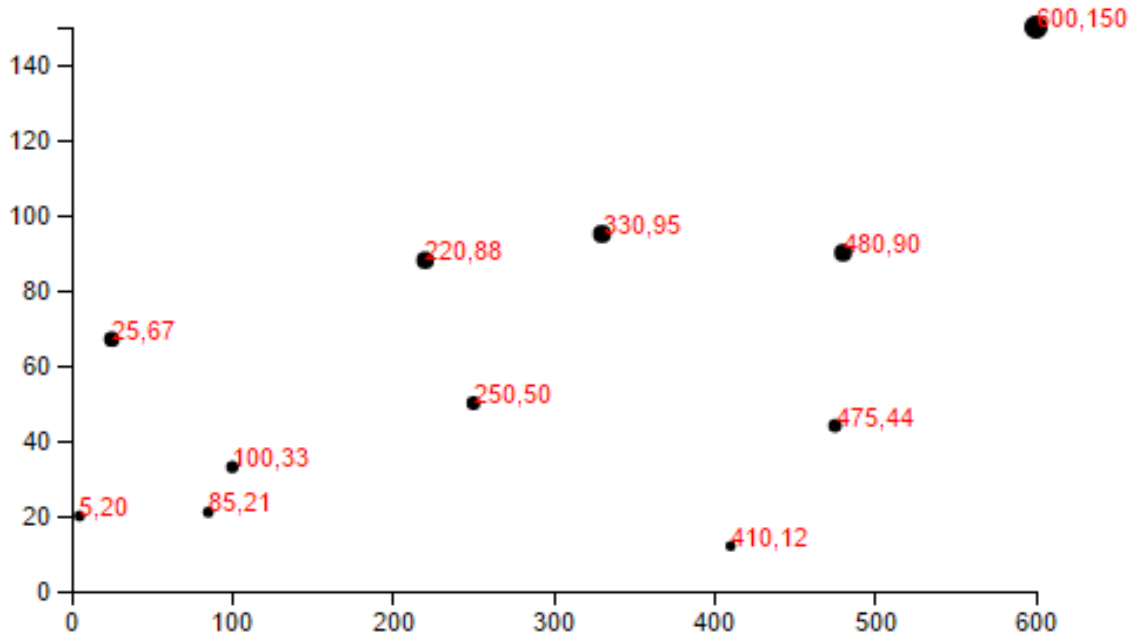


Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Ajout des Axes à Graphe

Y AXIX



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

updates permet de gérer les changement de données pour D3.

Les changements sont fait avec *transitions*, *motion*.

Mises à jour, Transitions, et Mouvement dans un Graphe

Echelle Ordinale

```
var xScale = d3.scale.ordinal()
```

```
> d3.range(10)
< [0, 1, 2, 3, 4, 5, 6, 7, 8,
  9]
>
```

```
.domain(d3.range(dataset.length))
```

On assigne à chaque donnée un ID qui correspond à sa position dans dataset.
Ceci est équivalent à:
.domain([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

.rangeBands([0, w])

Calcul des intervalles de 0 à w, et associer ces rangs aux intervalles.

(w - 0) / xScale.domain().length
(600 - 0) / 20
30

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.1.3 Design et Interaction guidés par les données

Mises à jour, Transitions, et Mouvement dans un Graphe

.rangeBands([0, w], 0.2)

20% de chaque intervalle est utilisé pour l'espacement.

.rangeRoundBands([0, w], 0.05);

Idem, avec un arrondi, 12.3456 devient 12.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Il existe de nombreux outils permettant de créer des visualisations de données : Tableau, Power BI, ChartBlocks.
Ces outils sont à utiliser sans écrire du code.
Comme alternative, il est possible d'utiliser la visualisation avec écriture du code: D3js présenté dans (2.1) et les librairies de Python .

Matplotlib, librairie de Python, est **complexe à utiliser**.
Une nouvelle bibliothèque basée sur Matplotlib : Seaborn a été développée.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 SEABORN [ref]

SEABORN: Plusieurs styles et palettes de couleur par défaut.

Les graphiques **relationnels**

Les graphiques **catégoriques**

Les graphiques de **distribution**

Les graphiques de **régression**.

[ref] https://seaborn.pydata.org/tutorial.html

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

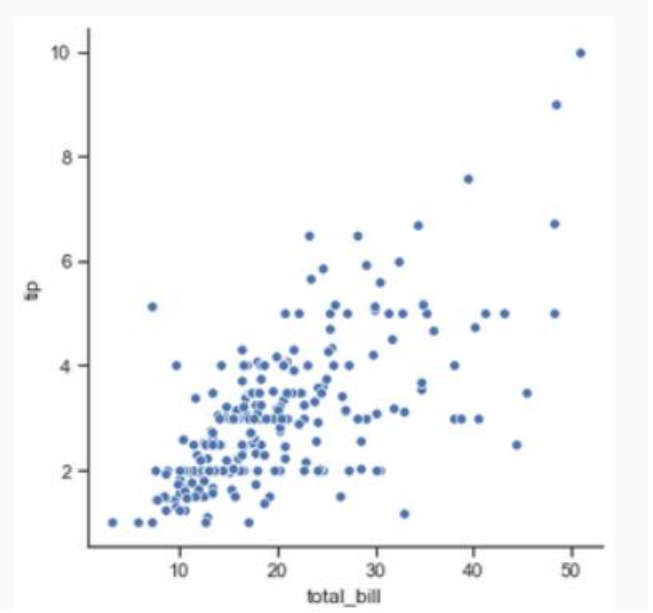
2.2 Librairies de Python

2.2.1 SEABORN

SEABORN: Plusieurs styles et palettes de couleur par défaut.

Les graphiques **relationnels** pour comprendre les relations entre deux variables: visualiser la relation statistique entre les points de données.

sns.relplot(…)



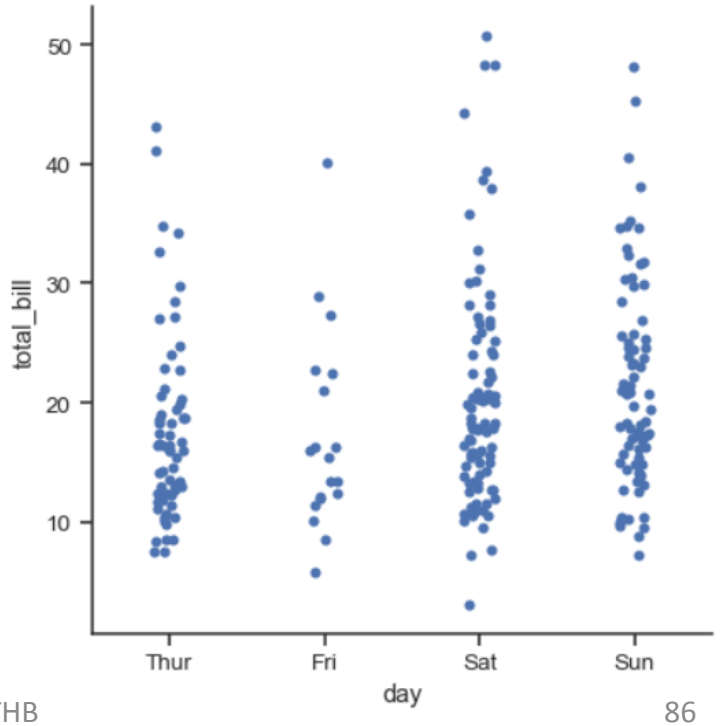
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 SEABORN

SEABORN: Plusieurs styles et palettes de couleur par défaut.

Les graphiques **catégoriques** pour visualiser des variables classées par catégorie.



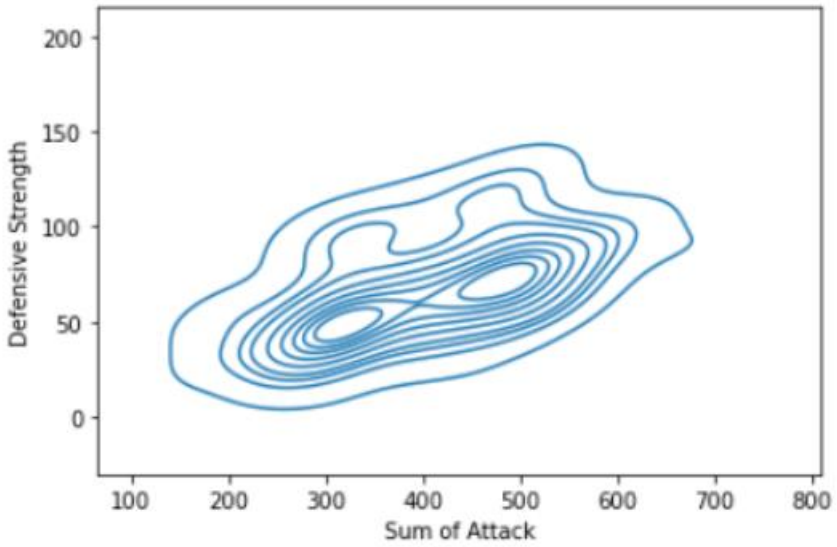
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 SEABORN

SEABORN: Plusieurs styles et palettes de couleur par défaut.

Les graphiques de **distribution** pour examiner les distributions uni-variées ou bi-variées.



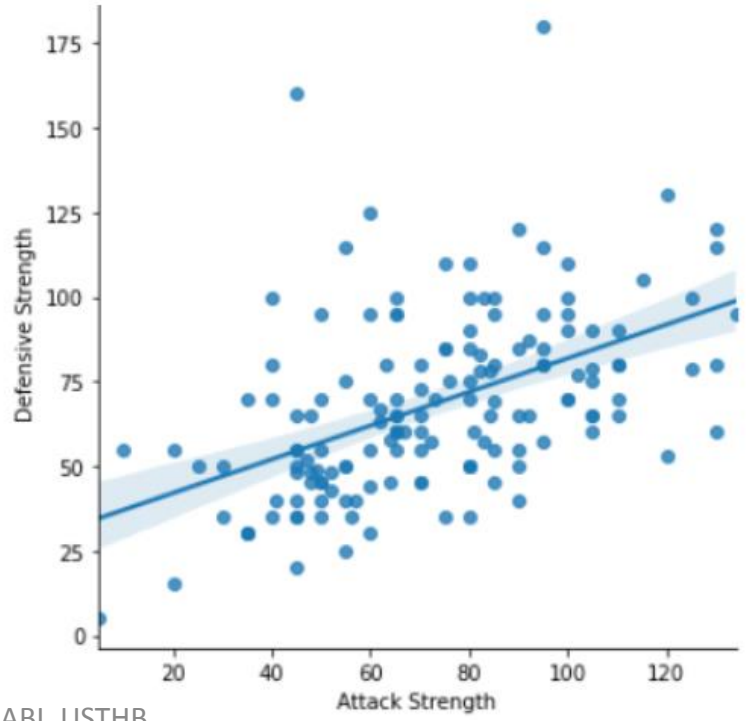
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 SEABORN

SEABORN: Plusieurs styles et palettes de couleur par défaut.

Les graphiques de **régression** pour ajouter un guide visuel des motifs dans un ensemble de données.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

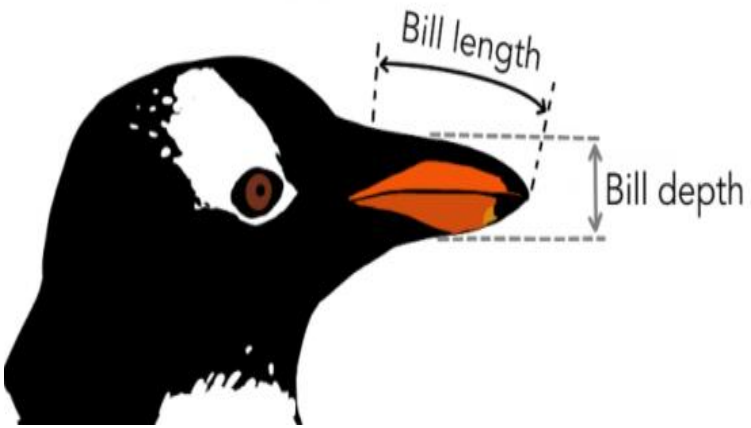
Univariate plot

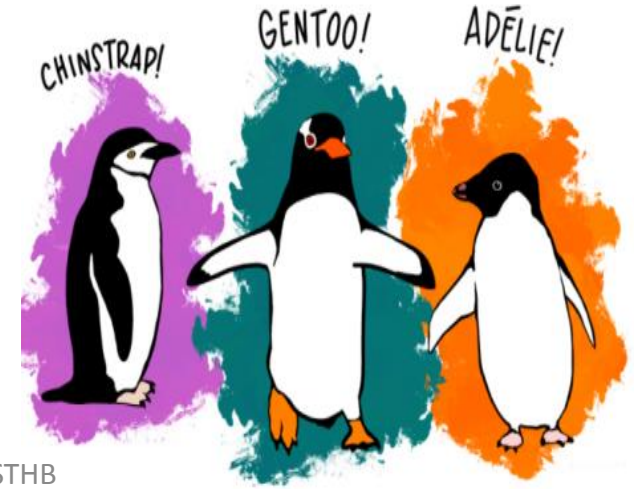
Le type par défaut de graphiques est l'histogramme:

le fichier penguins.csv:

species	island	bill_length_mm	bill_depth_mm	flipper_length	body_mass_g	sex
Adelie	*Torge*	*39.1*	*18.7*	*181*	*3750*	*male*
Adelie	*Torge*	*39.5*	*17.4*	*186*	*3800*	*female*
Adelie	*Torge*	*40.3*	*18*	*195*	*3250*	*female*

..





Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

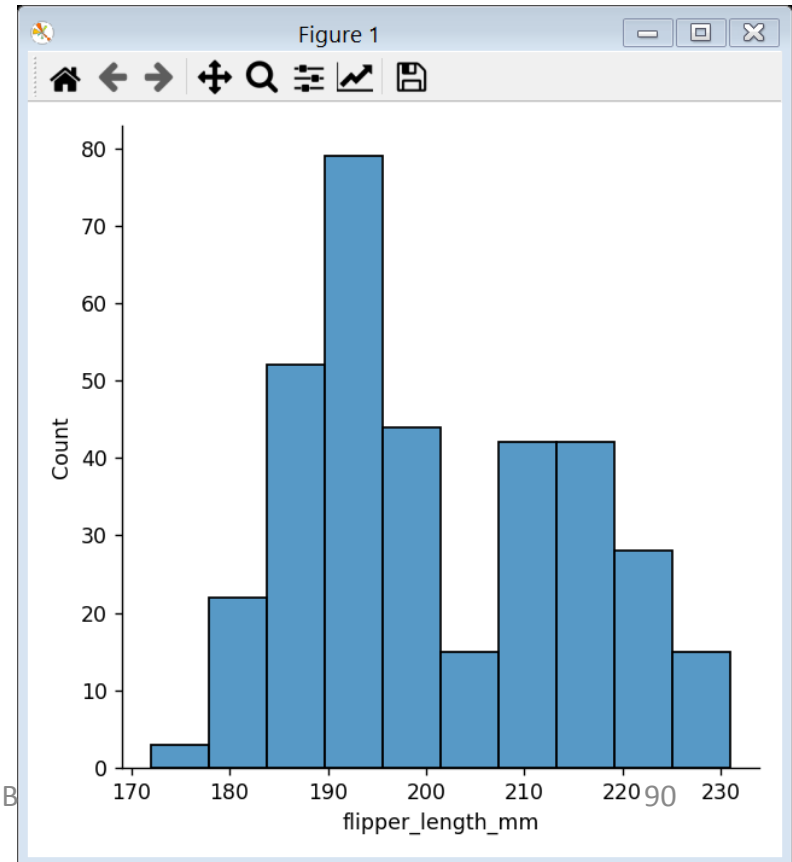
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm")
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

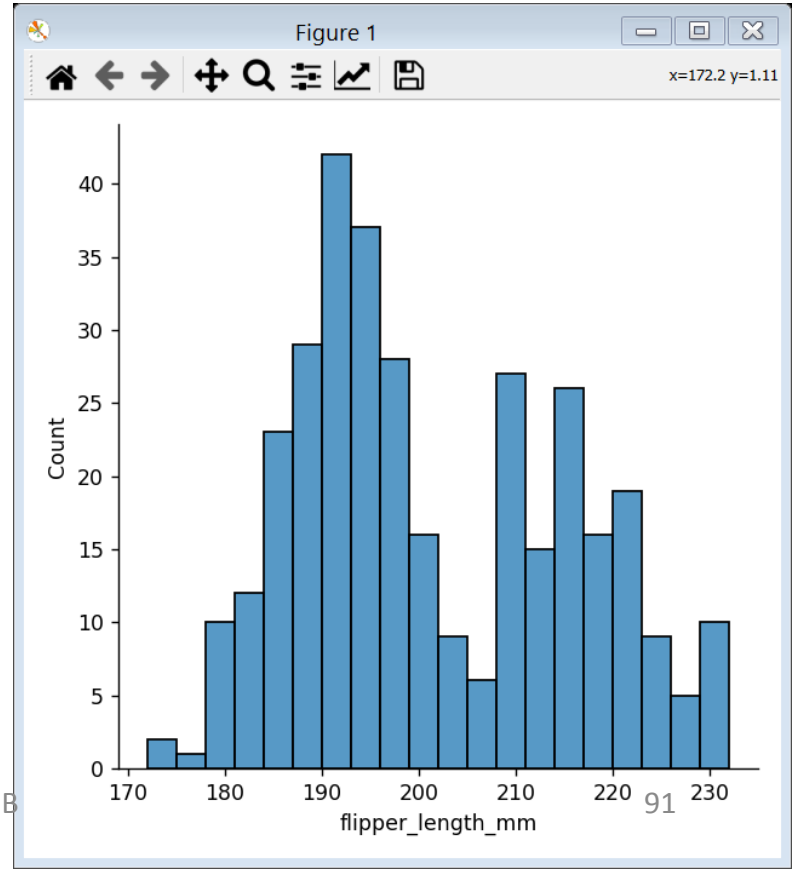
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm“,
binwidth=3)
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

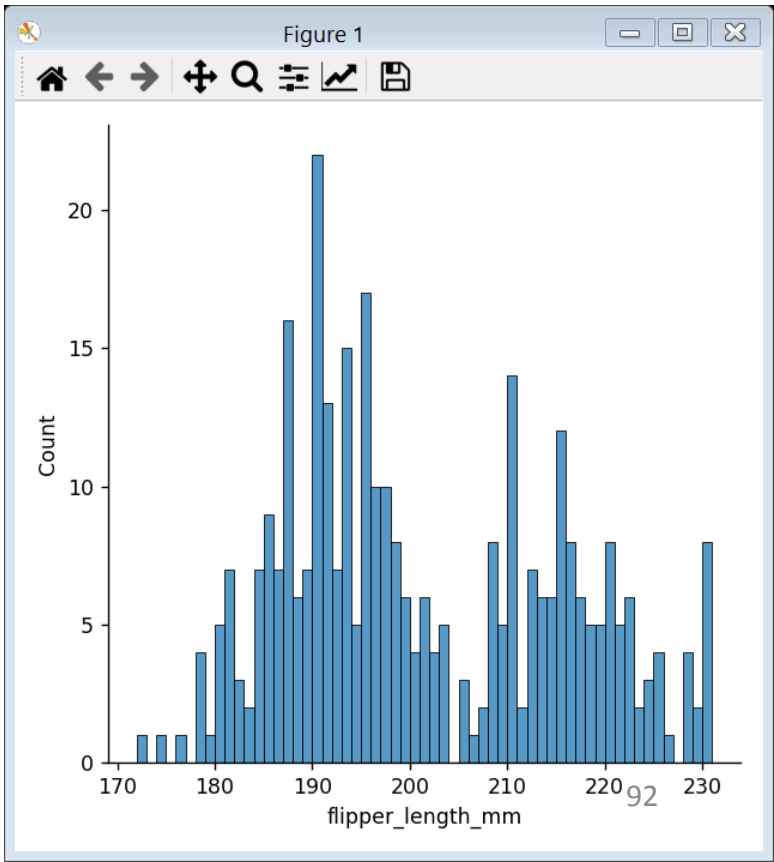
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm“,
binwidth=1)
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

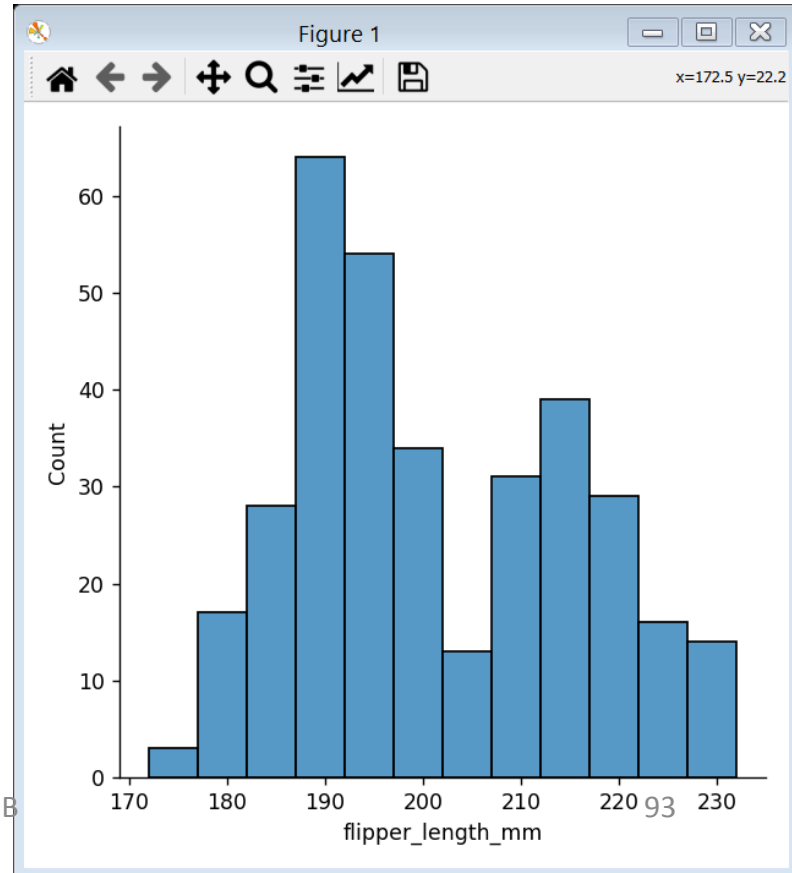
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm“,
binwidth=5)
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

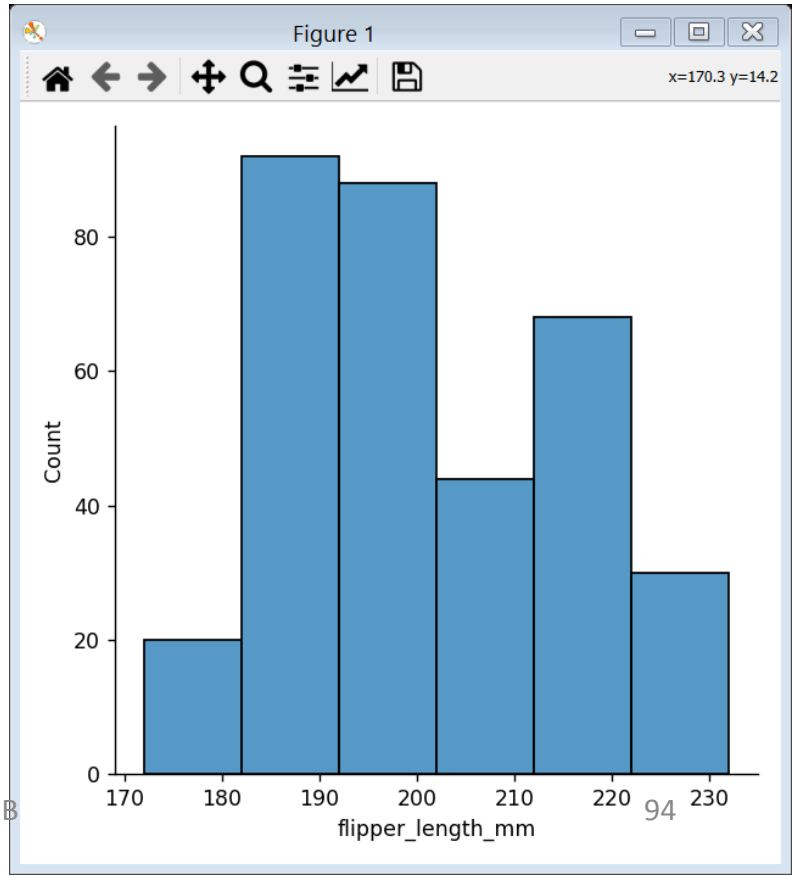
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm“,
binwidth=10)
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

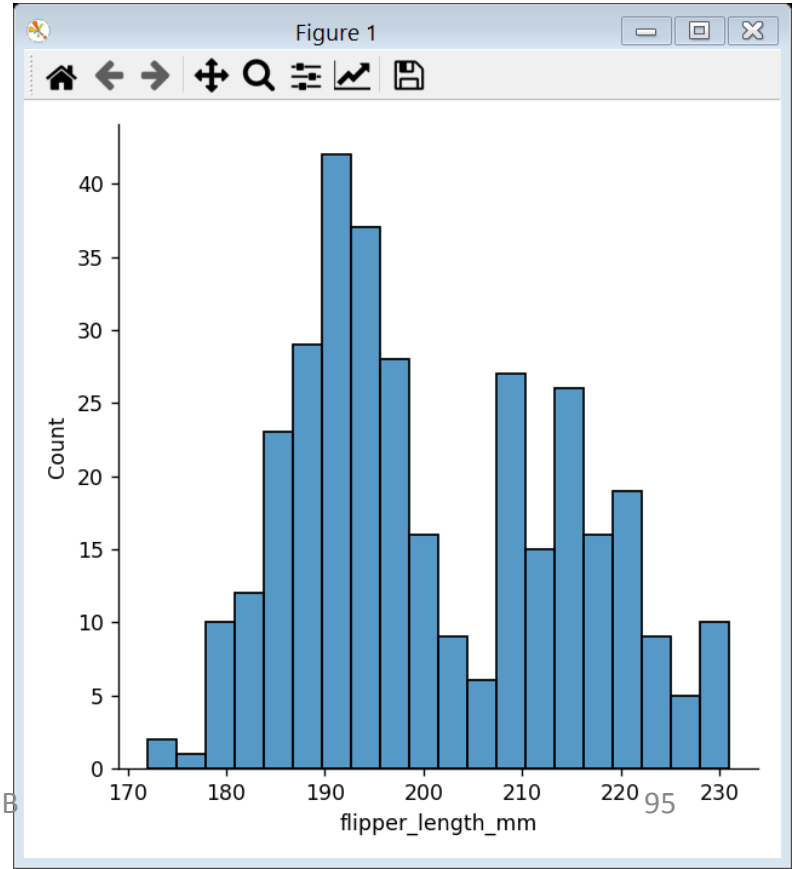
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm“, bins=20)
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

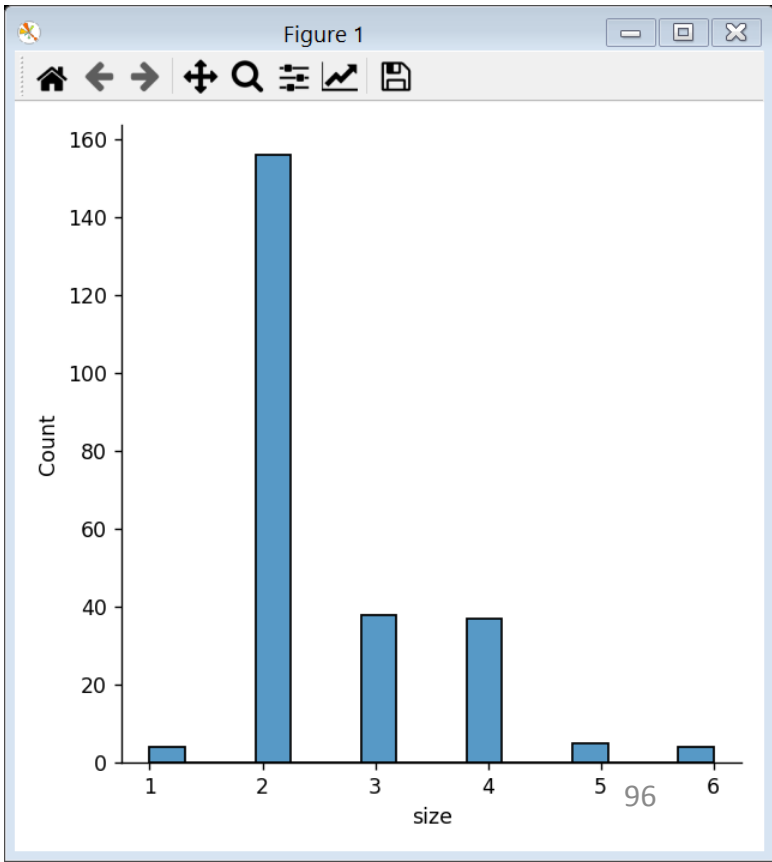
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

tips = sns.load_dataset("tips")
sns.displot(tips, x="size")
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

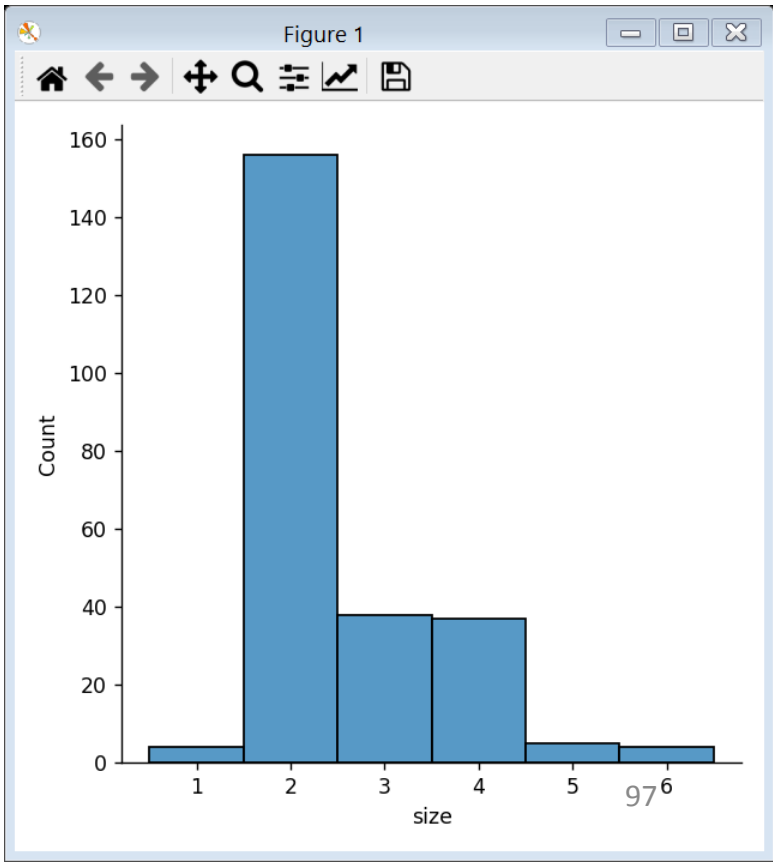
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

tips = sns.load_dataset("tips")
sns.displot(tips, x="size“, discrete=True)
```



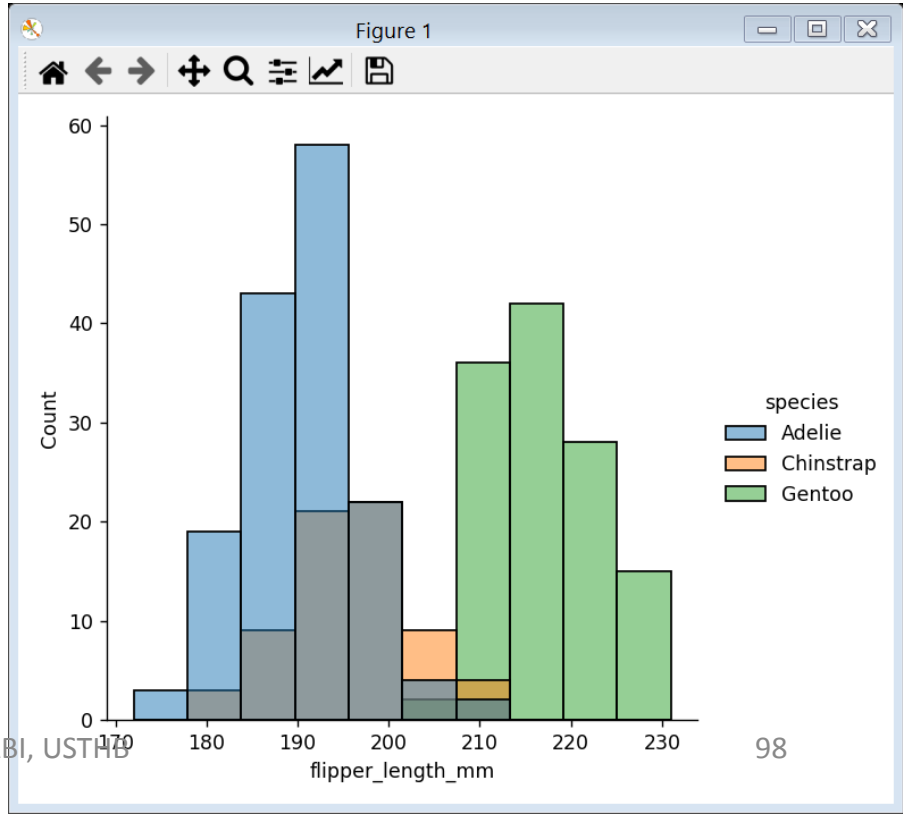
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:
L'attribution d'une variable « hue » dessinera un histogramme séparé pour chacune de ses valeurs uniques et les distinguera par couleur.




```
#import seaborn as sns

sns.displot(penguins, x="flipper_length_mm",
hue="species")
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

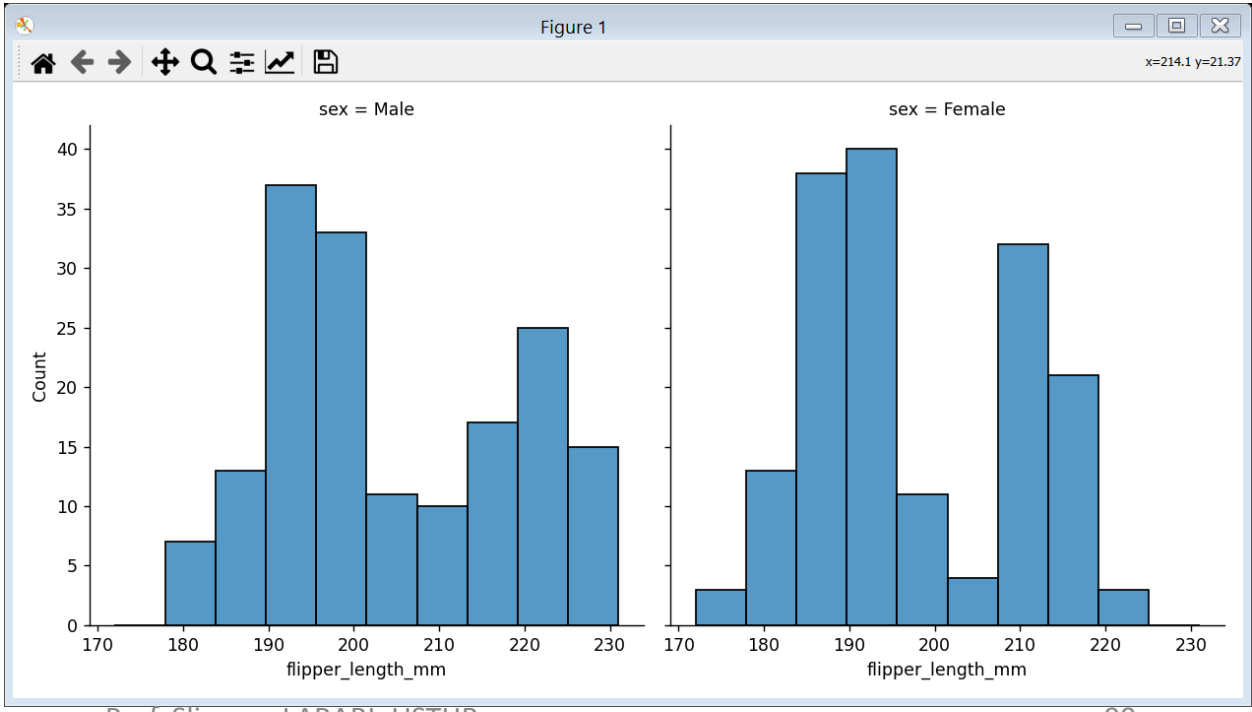
2.2.1 Seaborn

Univariate plot

Le type par défaut de graphiques est l'histogramme:

```
#import seaborn as sns

sns.displot(penguins,
x="flipper_length_mm",
col="sex")
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

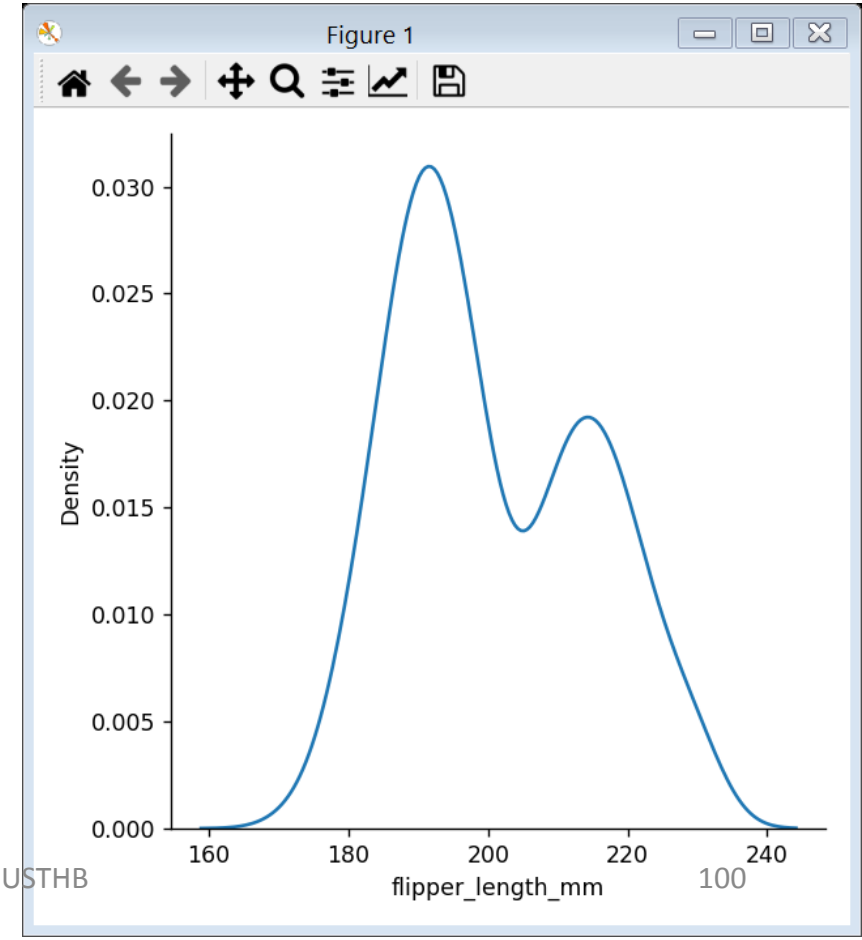
Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm"",
kind="kde"))
```

kde: Kernel Density Estimate.

Le tracé KDE lisse les observations avec un noyau gaussien, produisant une estimation de densité continue.



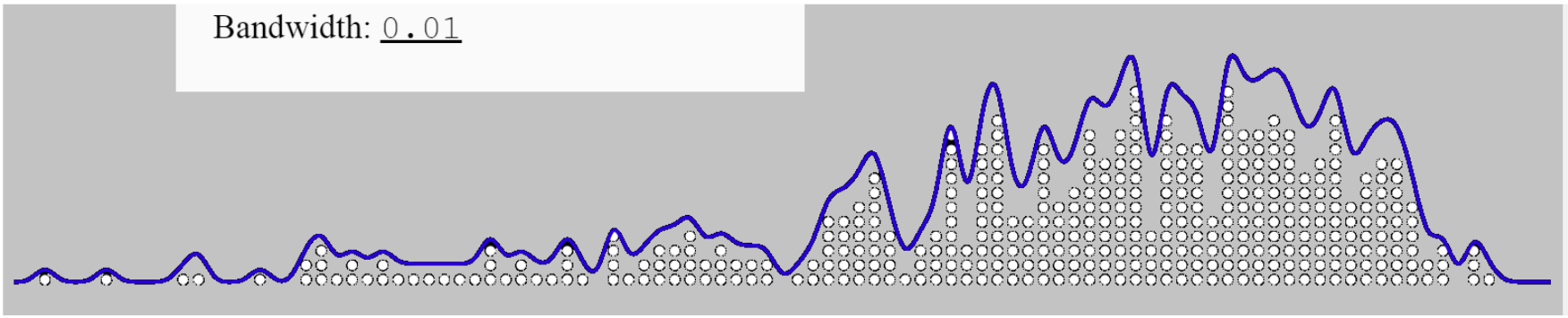
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

The KDE algorithm takes a parameter, *bandwidth*, that affects how "smooth" the resulting curve is.



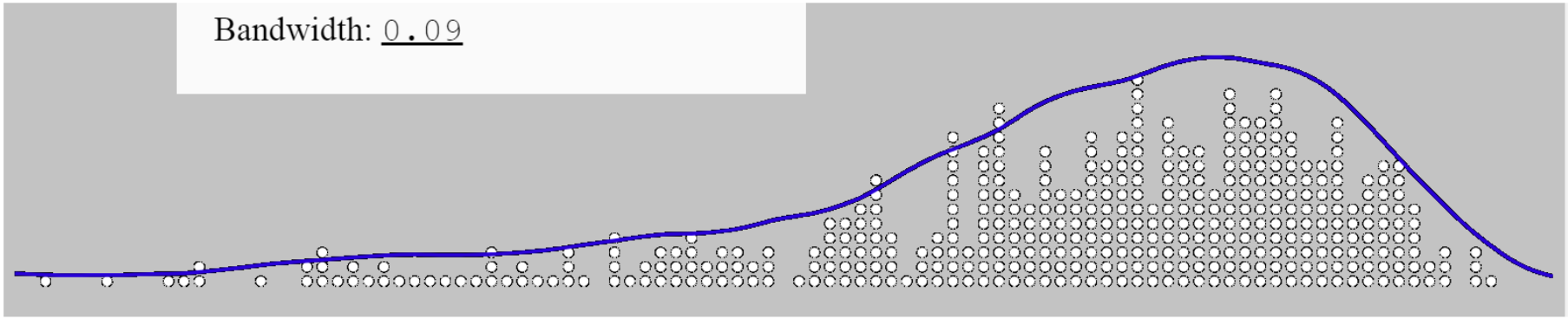
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

The KDE algorithm takes a parameter, *bandwidth*, that affects how "smooth" the resulting curve is.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

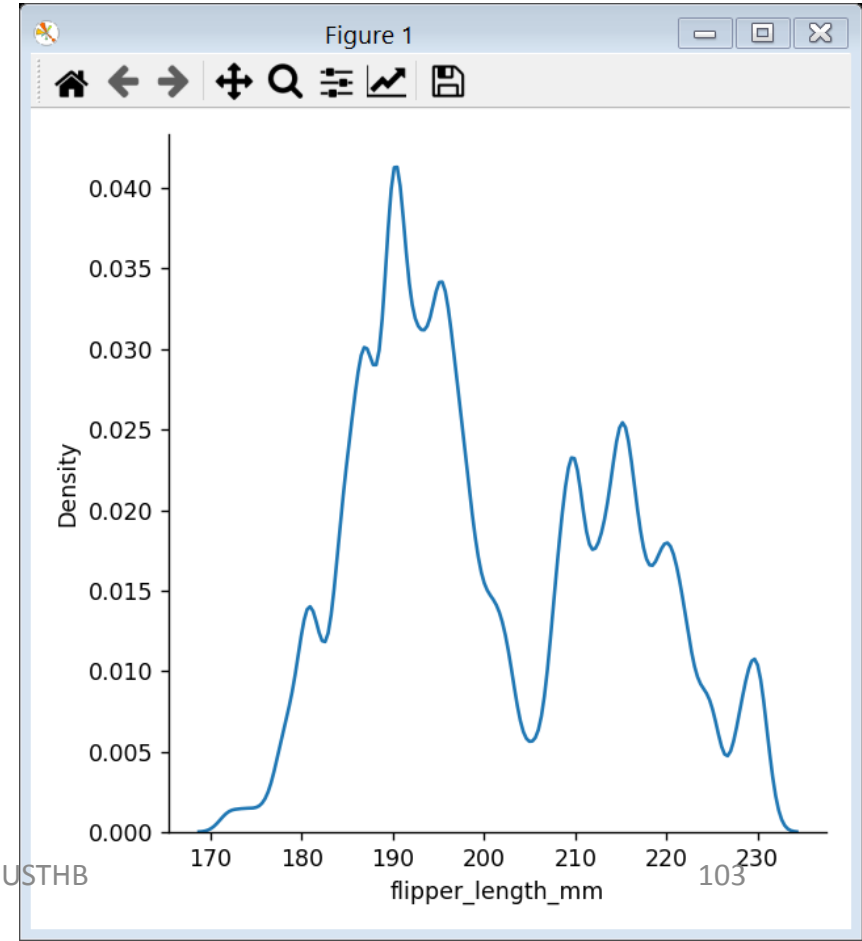
2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm"",
kind="kde“, bw_adjust=.25)
```

The KDE algorithm takes a parameter, *bandwidth*, that affects how “smooth” the resulting curve is.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

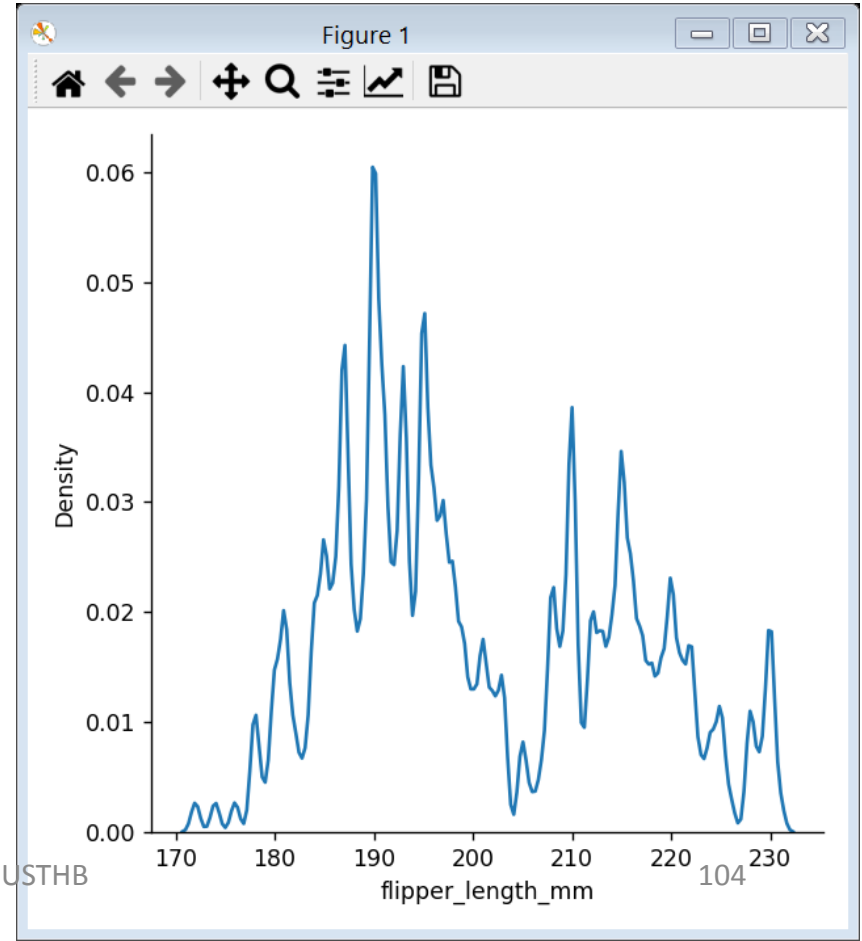
2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm"",
kind="kde“, bw_adjust=.1)
```



kde: Generate Kernel Density Estimate plot using Gaussian kernels.

Estimation non paramétrique la fonction de densité de probabilité d'une variable aléatoire pour un ensemble de données.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

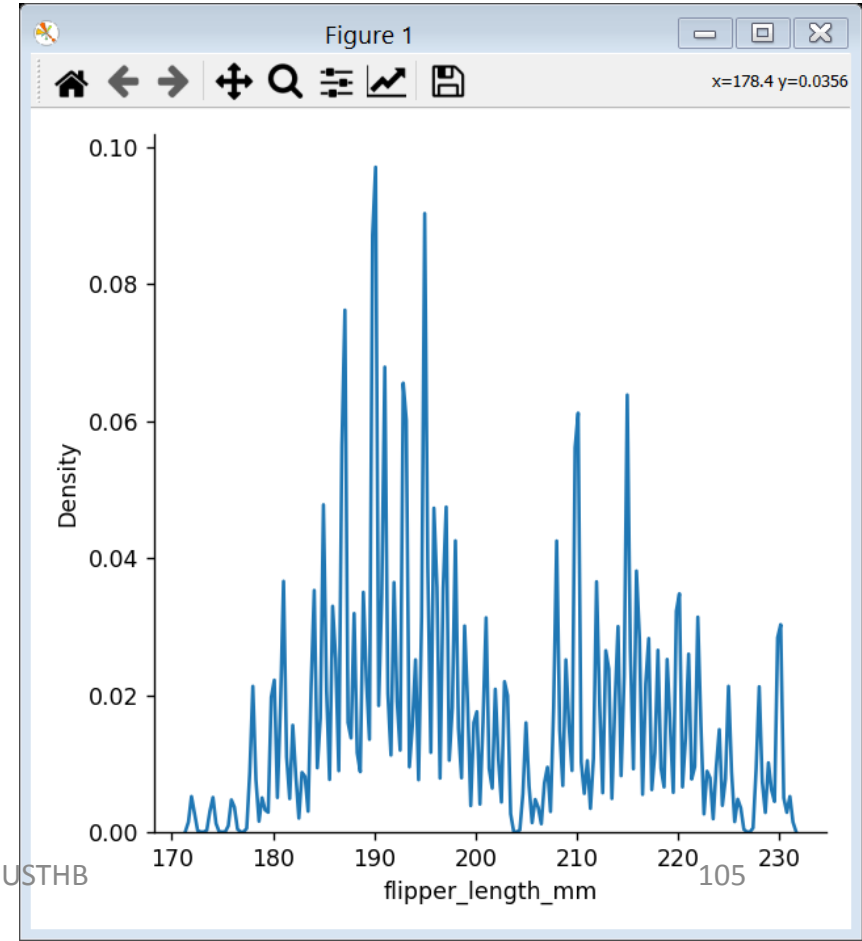
2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm"",
kind="kde“, bw_adjust=0.05)
```



kde: Generate Kernel Density Estimate plot using Gaussian kernels.

Estimation non paramétrique la fonction de densité de probabilité d'une variable aléatoire pour un ensemble de données.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

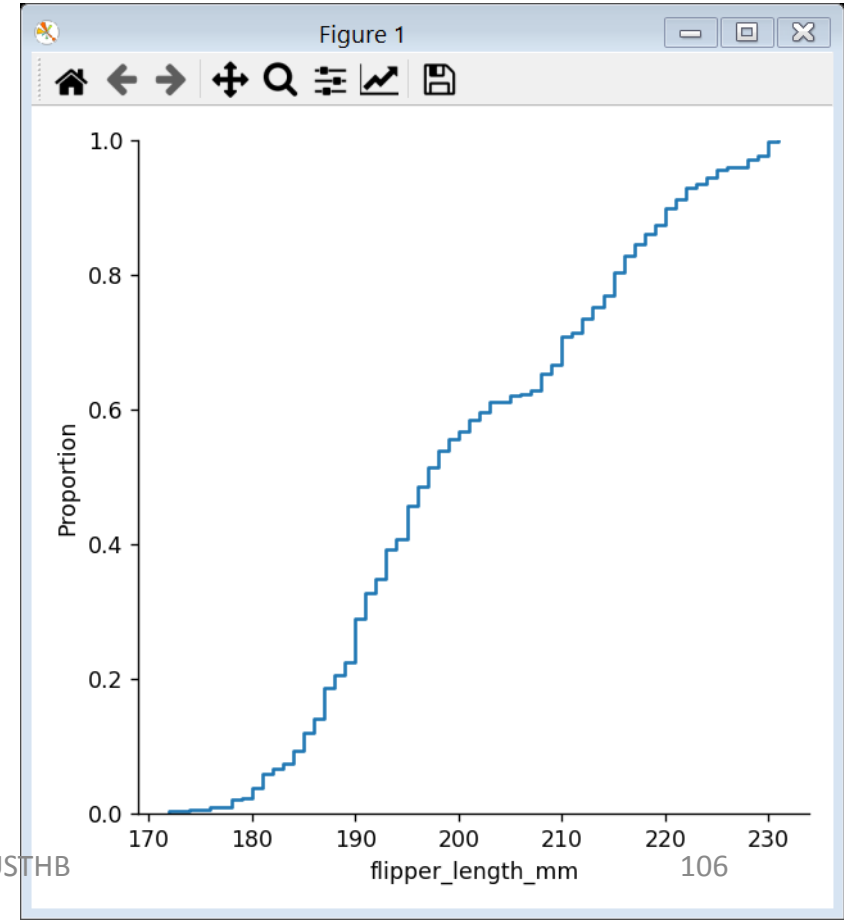
Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

```
#import seaborn as sns

penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm"", 
kind="ecdf"))
```

Ecdf: empirical cumulative distribution functions (ECDFs)

La fonction de répartition empirique (distribution cumulative) est une fonction qui attribue à chaque nombre réel x la proportion de valeurs d'échantillon inférieures ou égales à x.



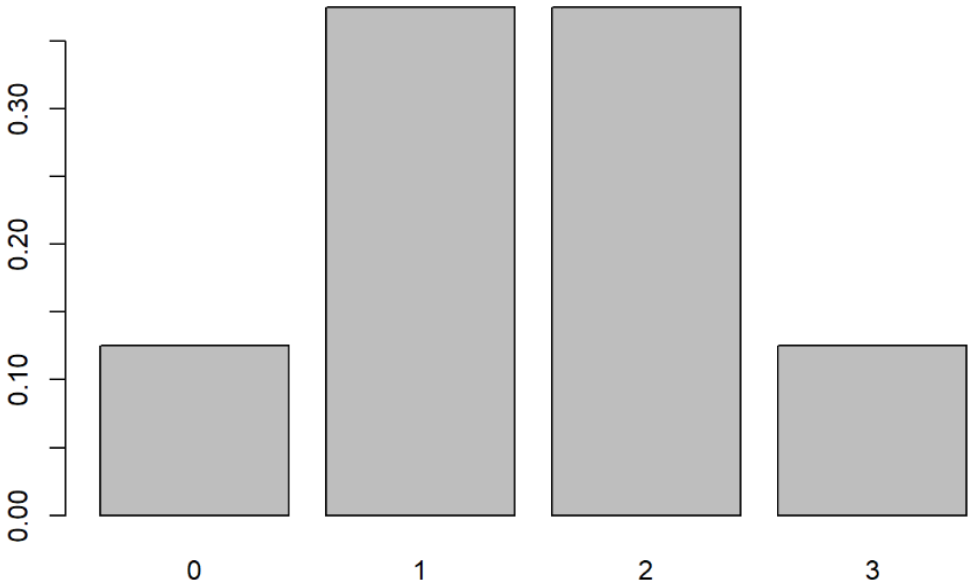
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

Exemple:
P(x=0)=1/8
P(x=1)= 3/8
P(x=2)= 3/8
P(x=3)= 1/8



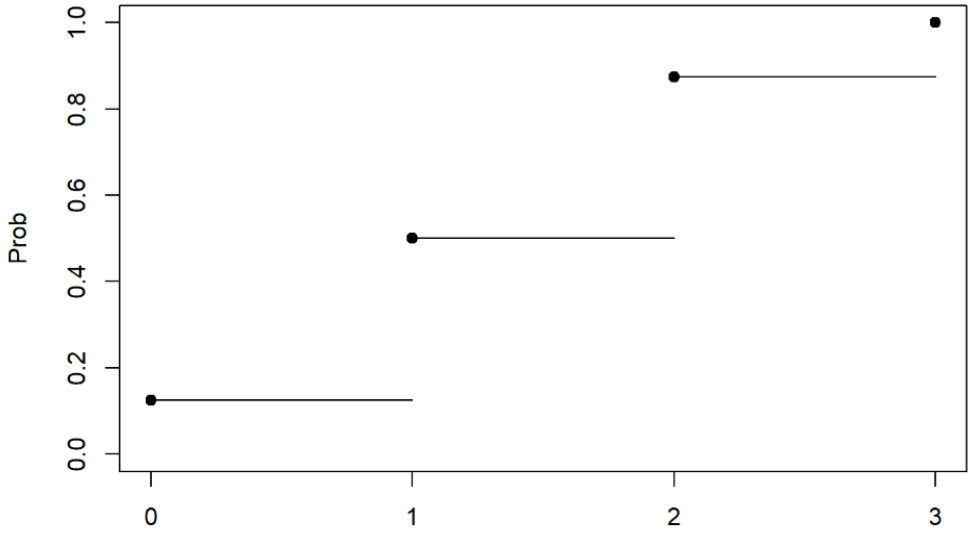
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Le paramètre **kind** est utilisé pour la sélection des différentes représentations:

Exemple:
P(x≤0)=1/8
P(x ≤ 1)= 4/8
P(x ≤ 2)= 8/8
P(x ≤ 3)= 1/8



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

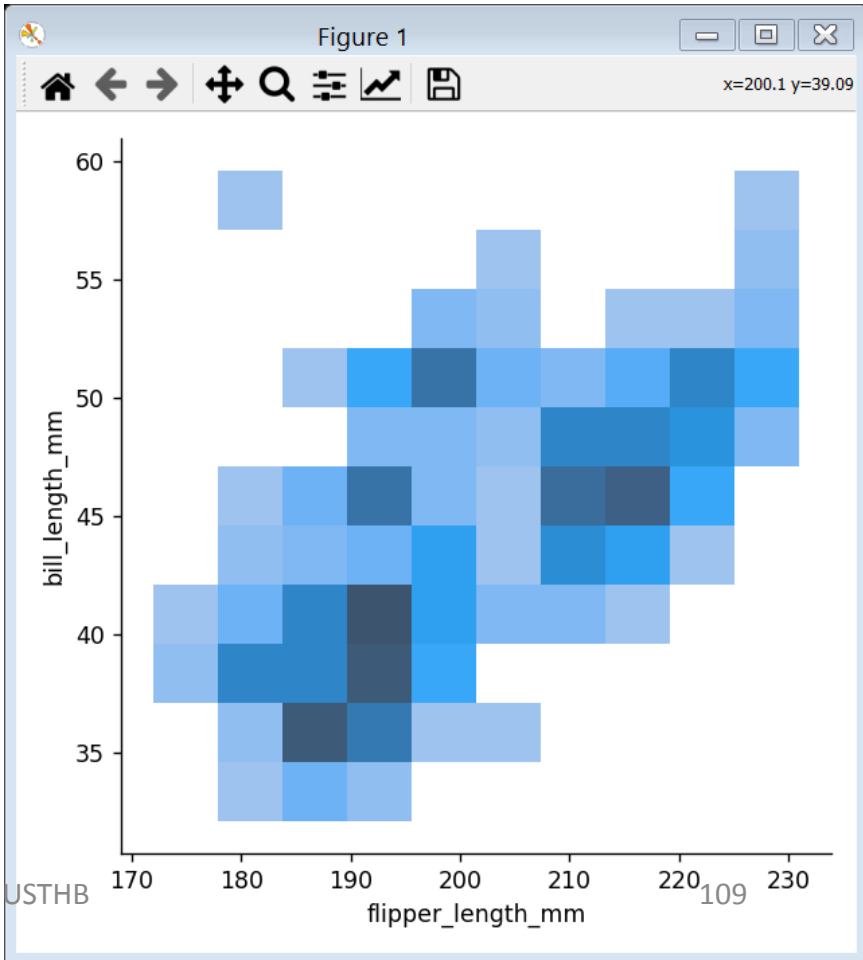
2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm")



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

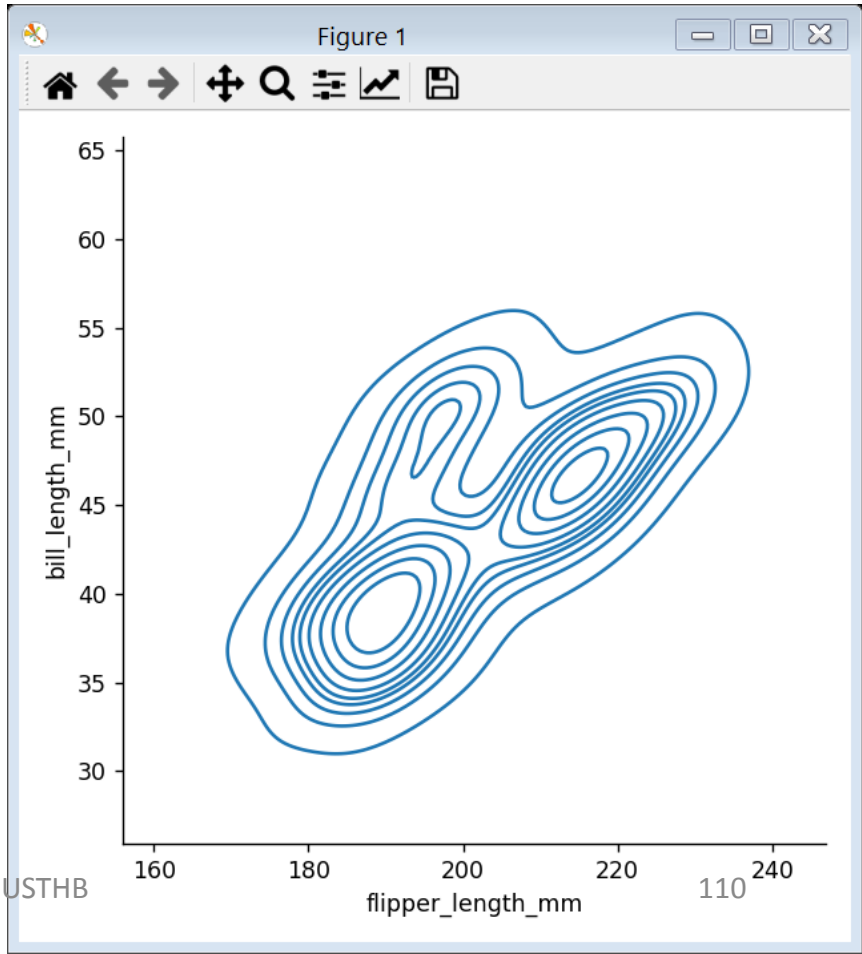
2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm“, kind=“kde”)

un tracé KDE bivarié lisse les observations (x, y) avec une gaussienne 2D.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

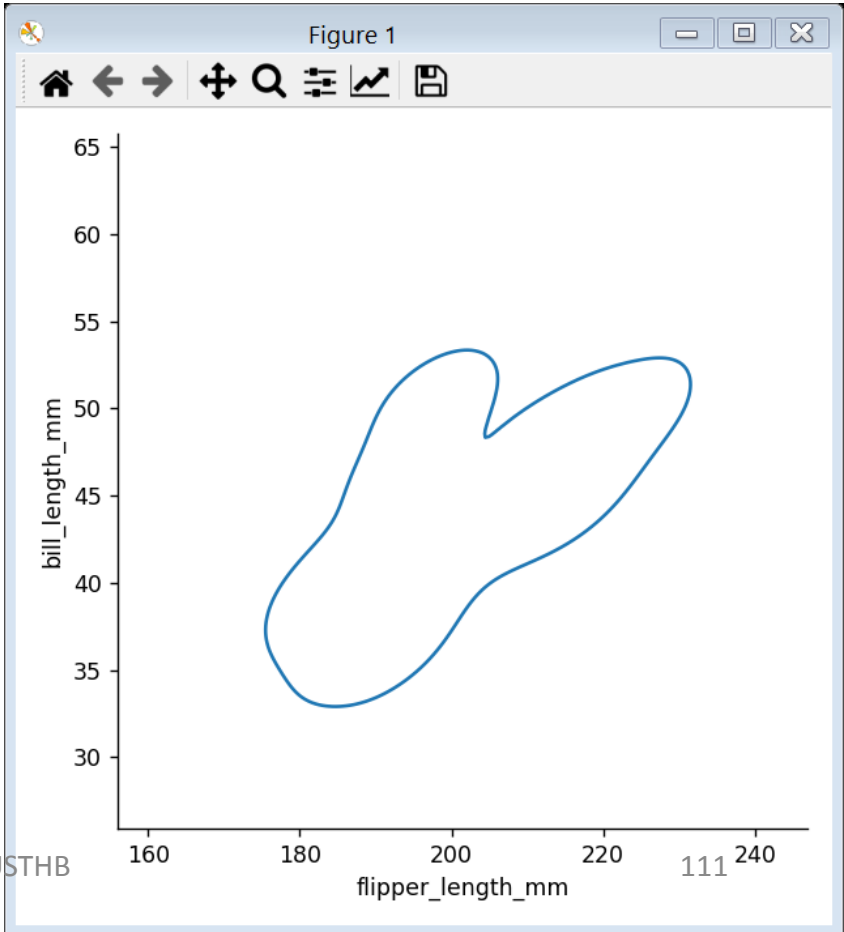
2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde", thresh=.2, levels=1)

Each curve shows a level set such that some proportion *p*(thresh) of the density lies below it.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

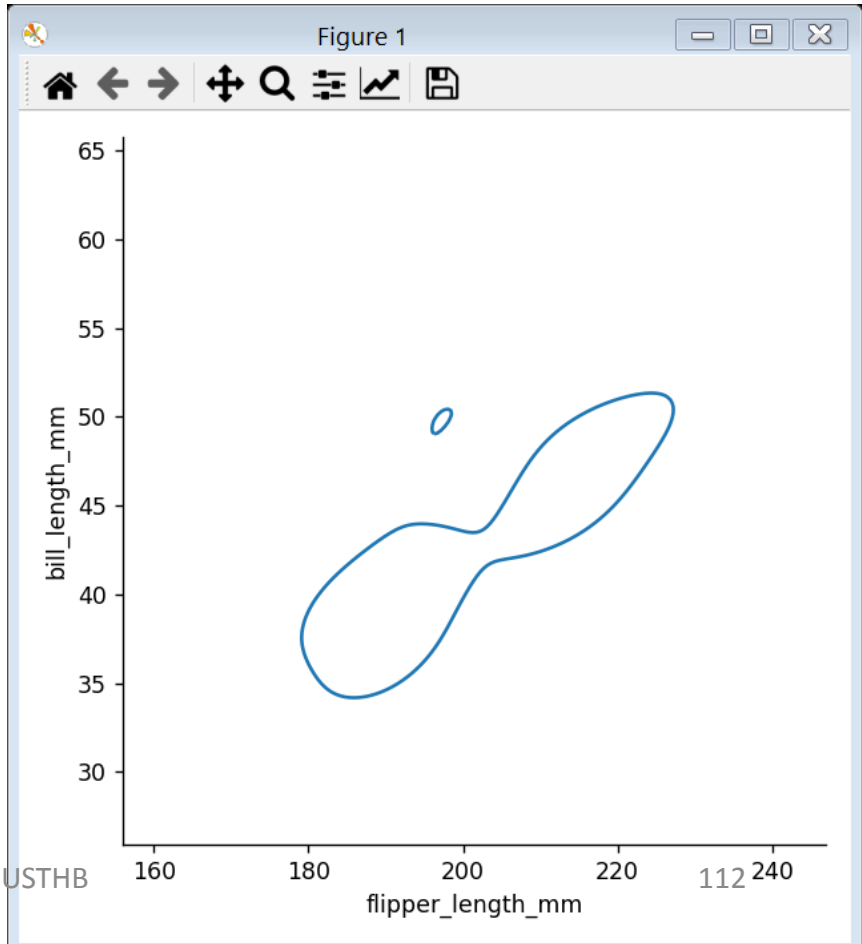
2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde", thresh=.5, levels=1)

Each curve shows a level set such that some proportion *p*(thresh) of the density lies below it.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

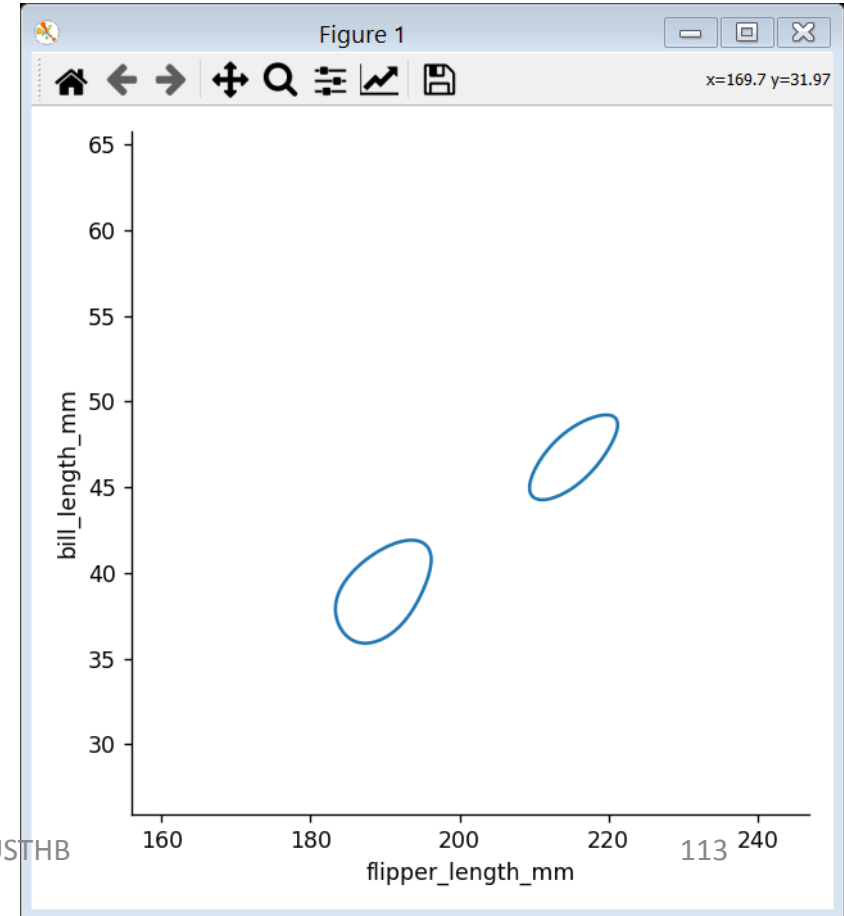
2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

```
sns.displot(data=penguins, x="flipper_length_mm",
y="bill_length_mm", kind="kde", thresh=.8, levels=1)
```



Each curve shows a level set such that some proportion *p*(thresh) of the density lies below it.

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

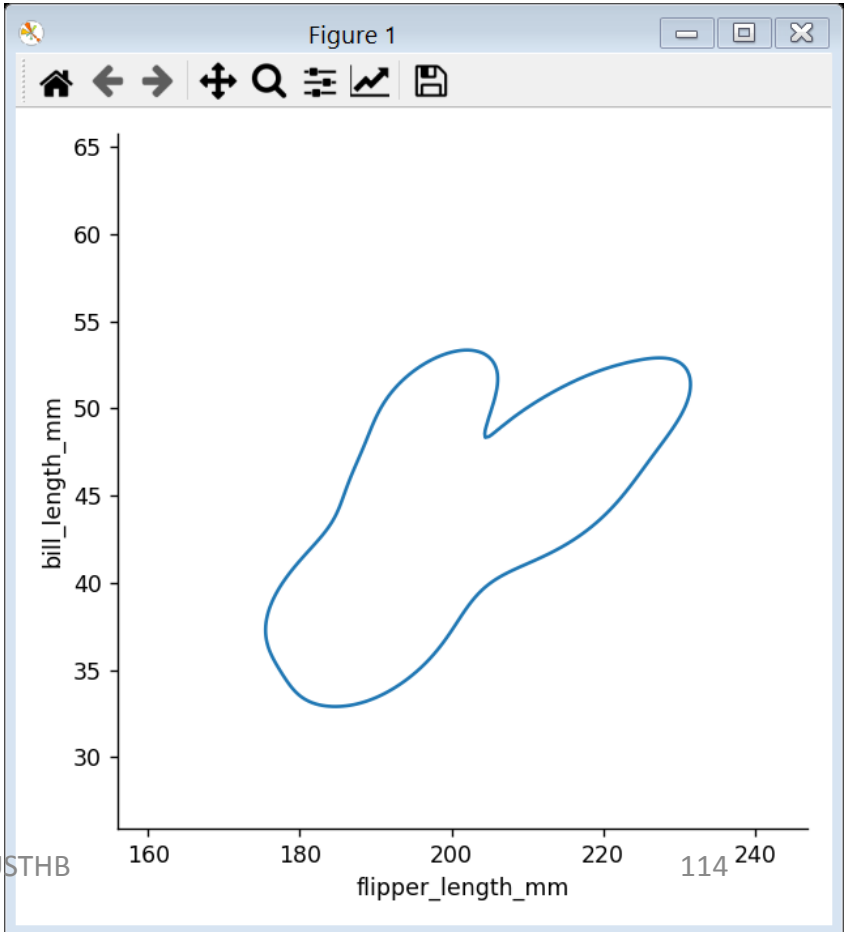
2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde", thresh=.2, levels=1)

Each curve shows a level set such that some proportion *p*(thresh) of the density lies below it.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

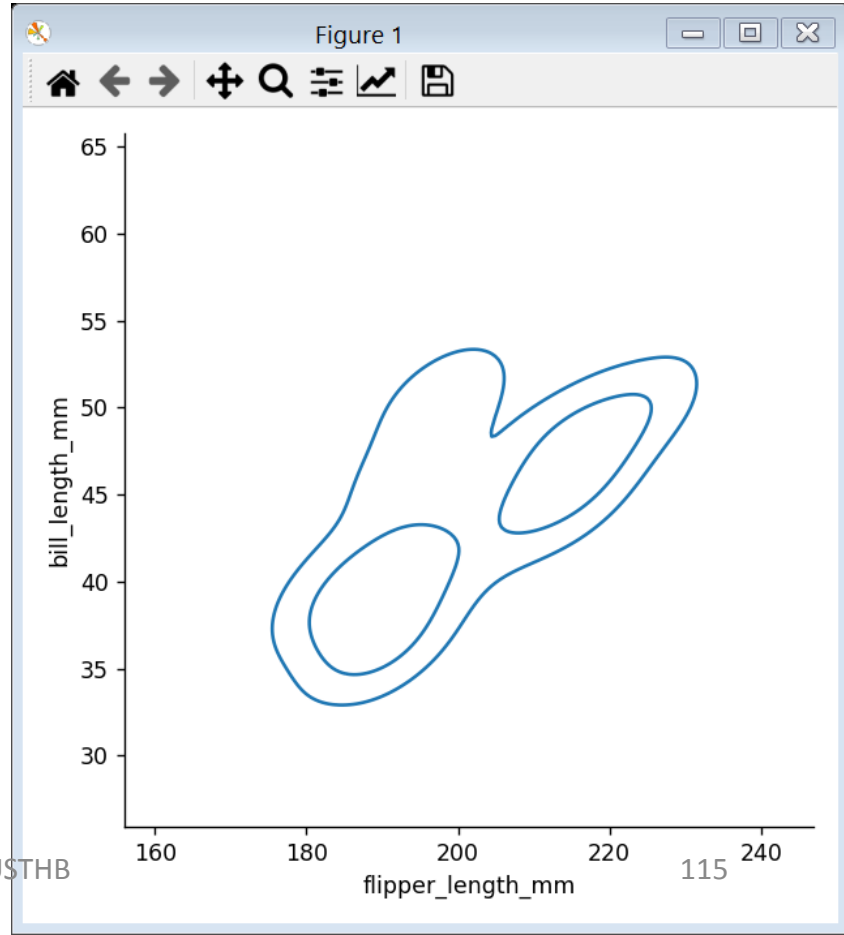
2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(data=penguins, x="flipper_length_mm", y="bill_length_mm", kind="kde", thresh=.2, levels=3)

Each curve shows a level set such that some proportion *p*(thresh) of the density lies below it.



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

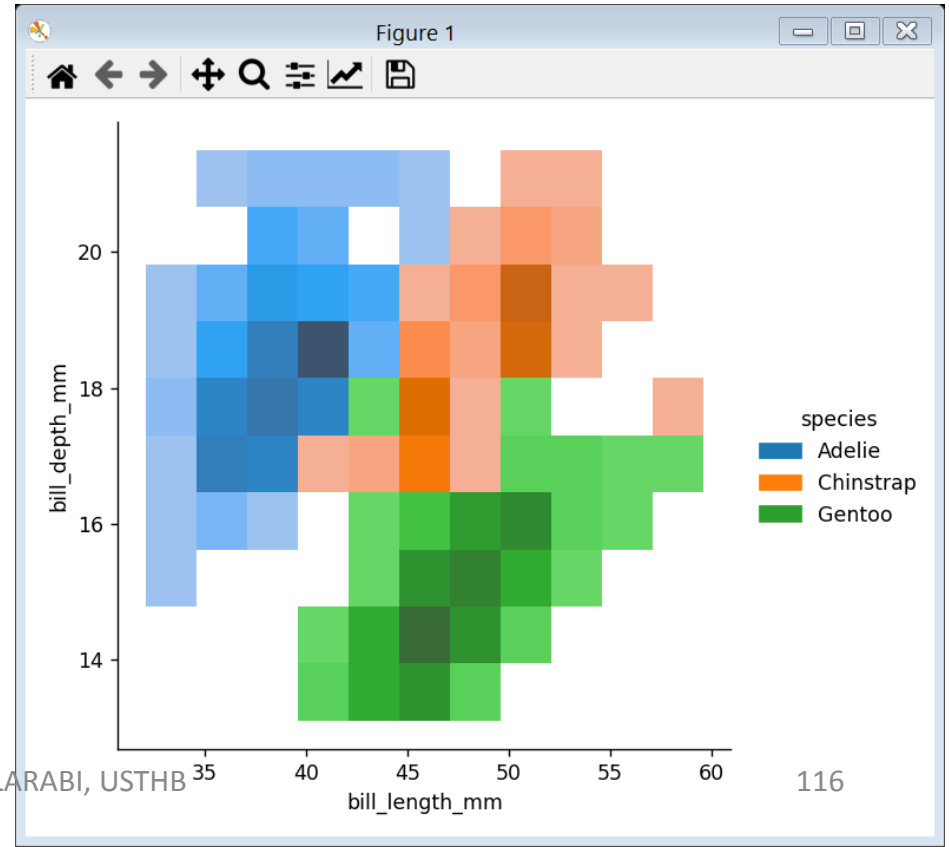
2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(penguins, x="bill_length_mm", y="bill_depth_mm", hue="species")



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

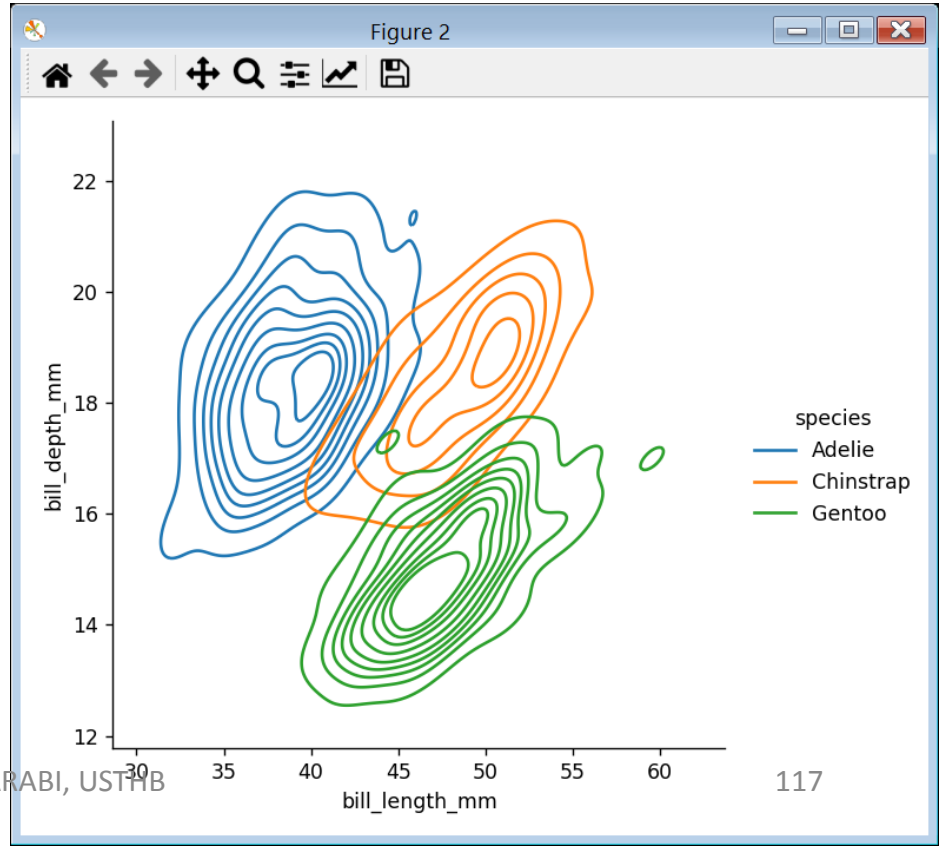
2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Disponible pour les types histogramme et KDEs

sns.displot(penguins, x="bill_length_mm", y="bill_depth_mm", hue="species“, kind="kde")



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

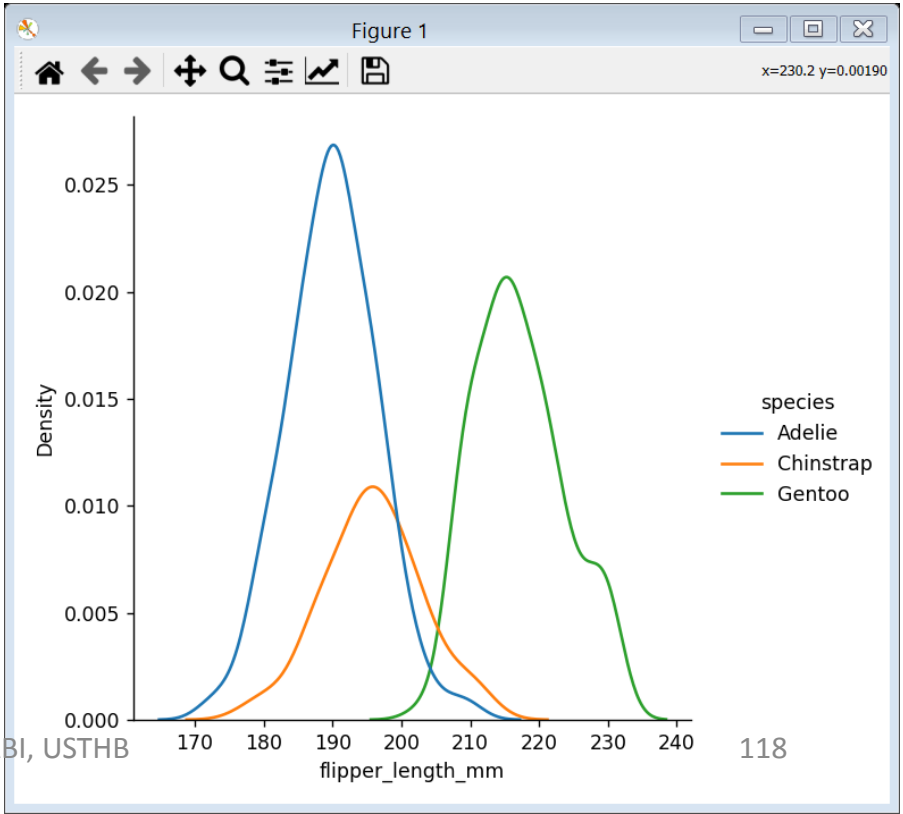
2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Chaque graphique peut être dessiné séparément pour des sous-ensembles de données à l'aide du mappage **hue**:

```
sns.displot(data=penguins,
x="flipper_length_mm", hue="species",
kind="kde")
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.1 Seaborn

Bivariate plot

Plotting many distributions

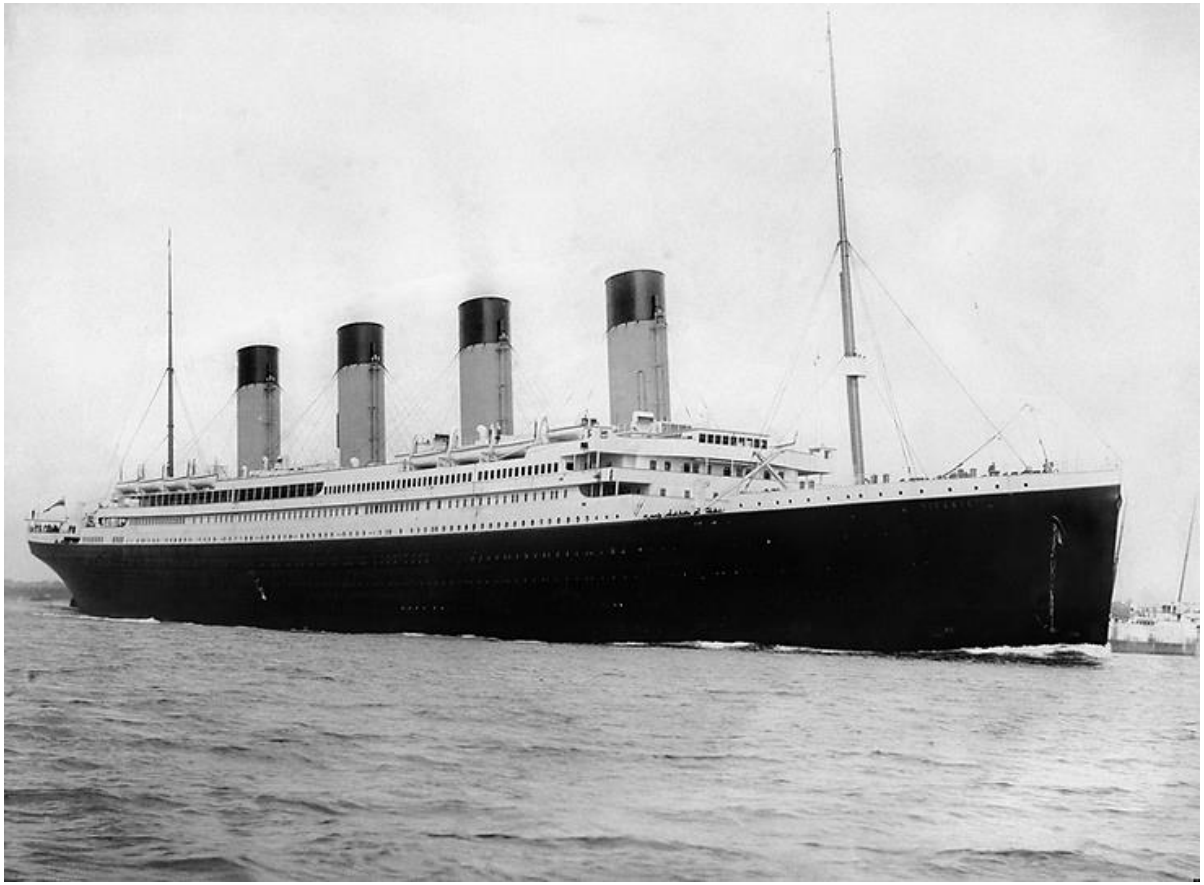
sns.pairplot(penguins)


Figure 1
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

- Importing the dataset and required libraries
- Data Cleaning(Dropping column with null values, statistical analysis)
- Data Visualization(Scatter Plot, Bar Plot)

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Data Discription:
survival: Survival (0 = No; 1 = Yes)
pclass: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name: Name
sex: Sex
age: Age
sibsp: Number of Siblings/Spouses Aboard
parch: Number of Parents/Children Aboard
ticket: Ticket Number
fare: Passenger Fare
cabin: Cabin
embarked: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Importing the Dataset and Required Libraries

```
import pandas as pd

# data processing, CSV file I/O (e.g. pd.read_csv)

import seaborn as sns
import matplotlib.pyplot as plt

#Importing the dataset using pandas read_csv

df= pd.read_csv('titanic-data.csv')
df.head()
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

A statistical summary of the imported dataset.

df.describe()

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

datatypes in the dataset

pandas.info() method

```
[8 rows x 7 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Check number of null values in a column

print(df.isnull().sum())

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

suppression de la colonne non utilisée et ayant un nombre maximal de valeurs nulles, c'est-à-dire la colonne Cabin.

```
df_cleaned = df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)

df_cleaned.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

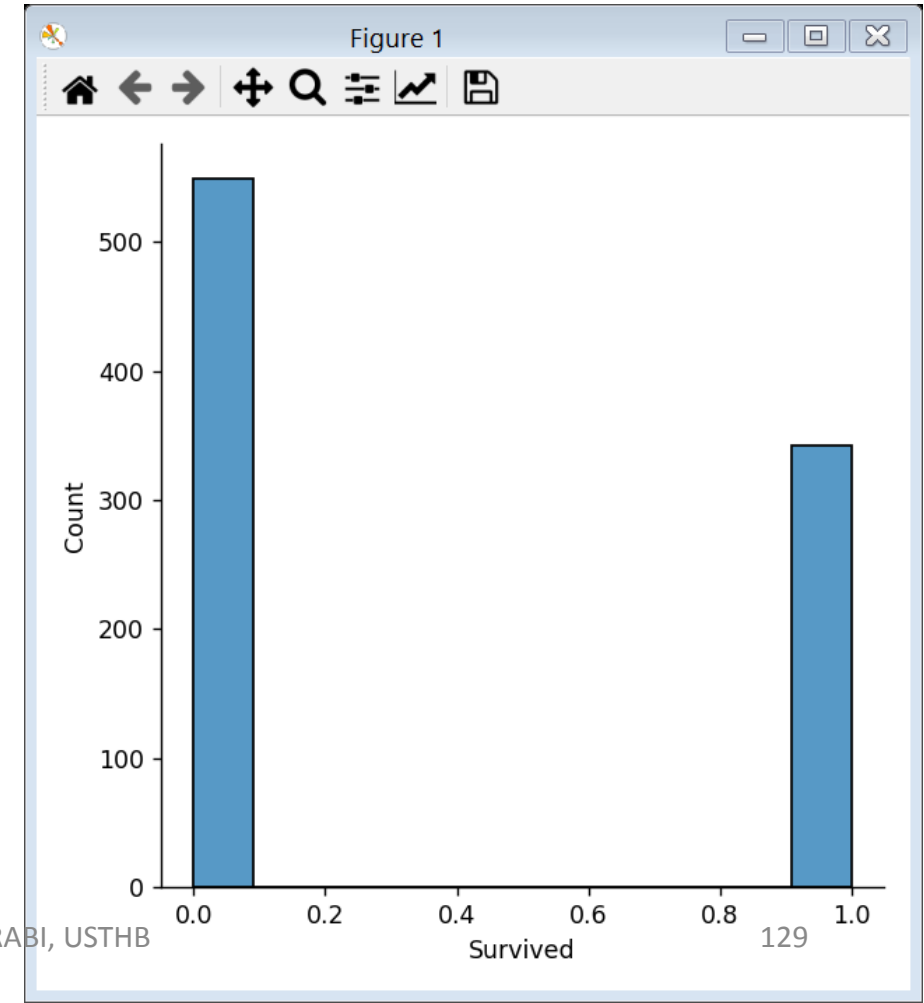
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Visualisation

```
sns.displot(data=df_cleaned, x="Survived")
plt.show()
```



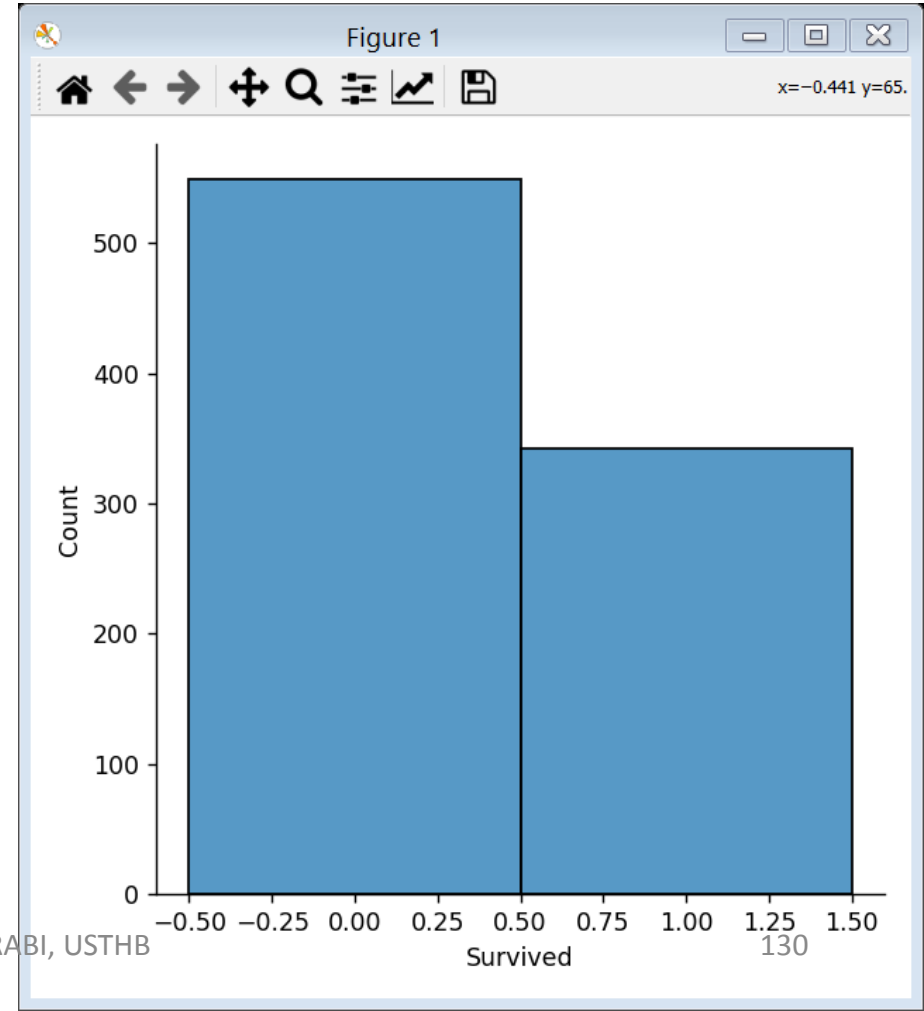
Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Visualisation

```
sns.displot(data=df_cleaned, x="Survived“,
discrete=True)
plt.show()
```



Chapitre 2. VISUALISATION DE DONNEES SPATIALES

2.2 Librairies de Python

2.2.2 Exemple de visualisation avec Seaborn: Titanic dataset

Visualisation

```
sns.displot(data=df_cleaned, x="Pclass",
discrete=True)
plt.show()
```