IIC2026
Visualización de Información

Hernán F. Valdivieso López
(2023 - 2 / Clase 19)

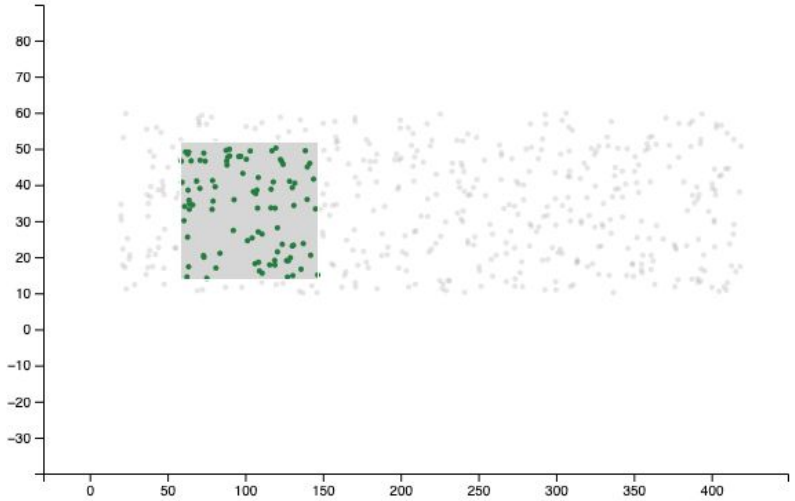
Temas de la clase - Brushing y agregación en D3

1. *Brush* en D3
2. Agregación de datos en D3
3. *Hexbin* en D3

Brush en D3

Brush en D3

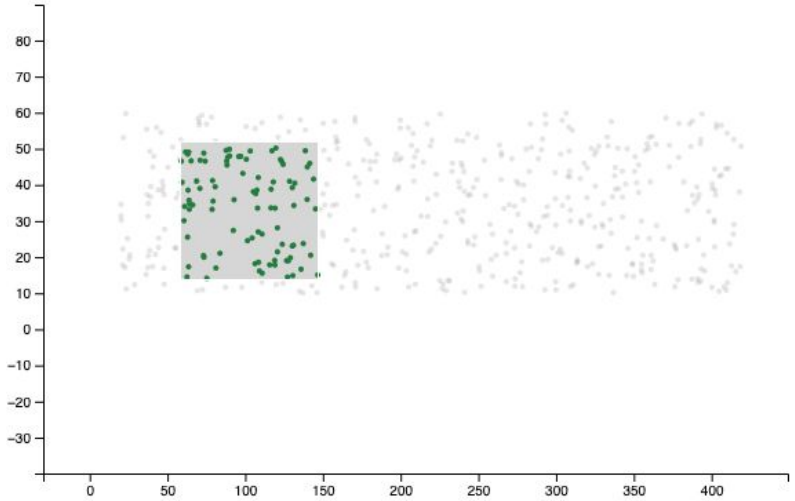
- El ***brush*** es una forma interactiva de seleccionar una región de la visualización de una o dos dimensiones mediante *click* y arrastrar el mouse.
- D3 provee de un método **d3.brush** encargado de gestionar todos los eventos relacionados a esta funcionalidad.
- Cuando se ocupa este método, D3 crearé los objetos SVG necesarios para visualizar y gestionar el brush.



Brush en D3

Cuando se ocupa este método, D3 creará los objetos SVG necesarios para visualizar y gestionar el brush.

- **1 `rect clase overlay`** → indica donde podemos hacer brush.
- **1 `rect clase selection`** → indica qué datos están seleccionados por el brush.
- **8 `rect clase handle`** → en los costados y equinas del `rect clase selection`. Permiten alterar el tamaño del cuadro de selección.



Brush en D3

Pasos para usar *brush*

1. Crear un contenedor para nuestro *brush* (crear un g)
2. Crear nuestro objeto *brush* indicando la región donde podrá actuar y la función a llamar cuando se gatille este evento.
3. **(opcional)** Definir un filtro de qué eventos específicos considerar.
4. Llenar ese contenido con nuestro objeto brush (`contenedor.call(brush)`)
5. **(opcional)** Alterar algún elemento creado por el brush.

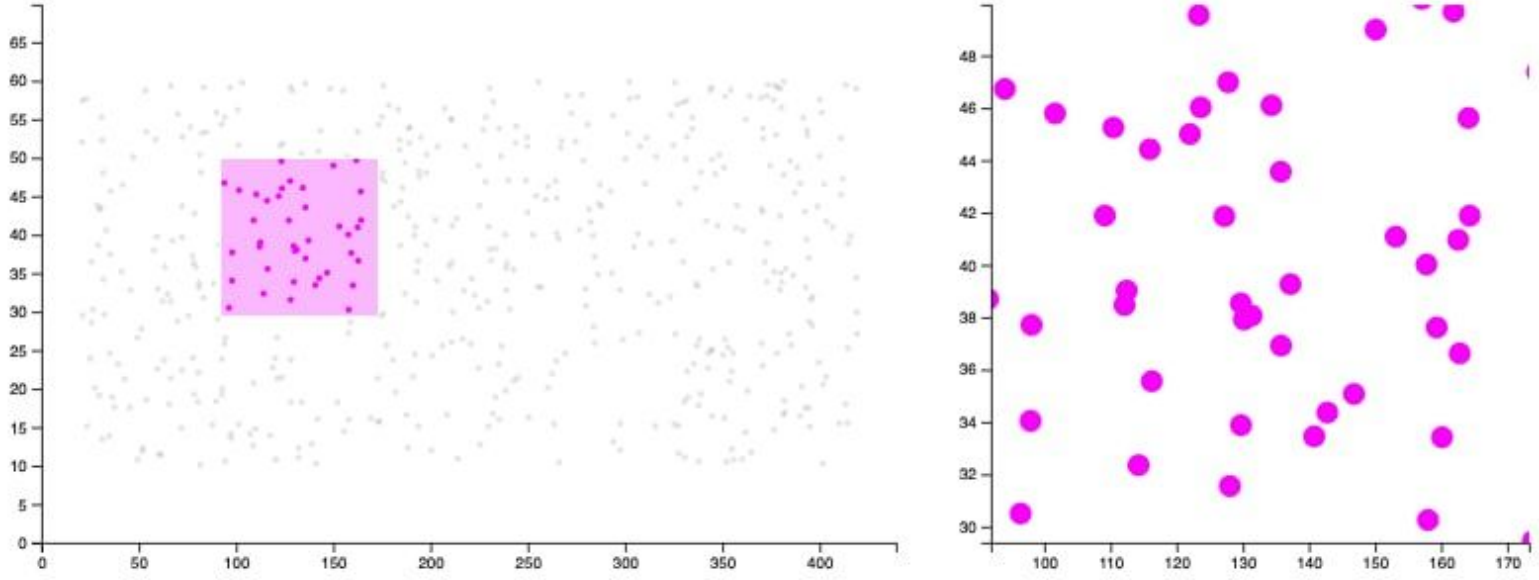
Brush en D3

Vamos al código 👨‍💻 👩‍💻

Brush en D3

¿Qué puedo hacer con Brush?

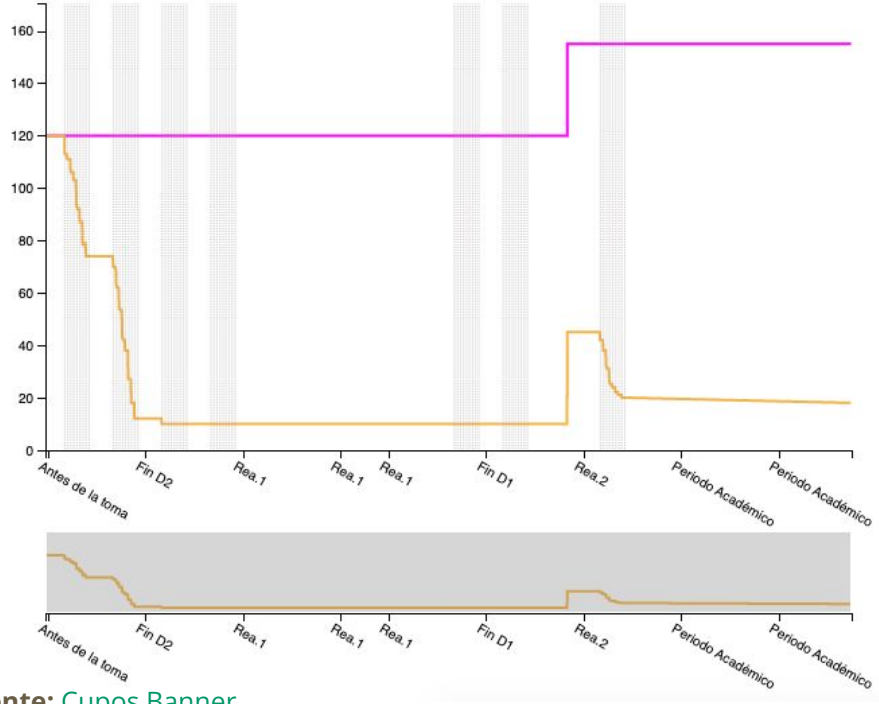
- **Una vista de minimapa y otra en detalle**

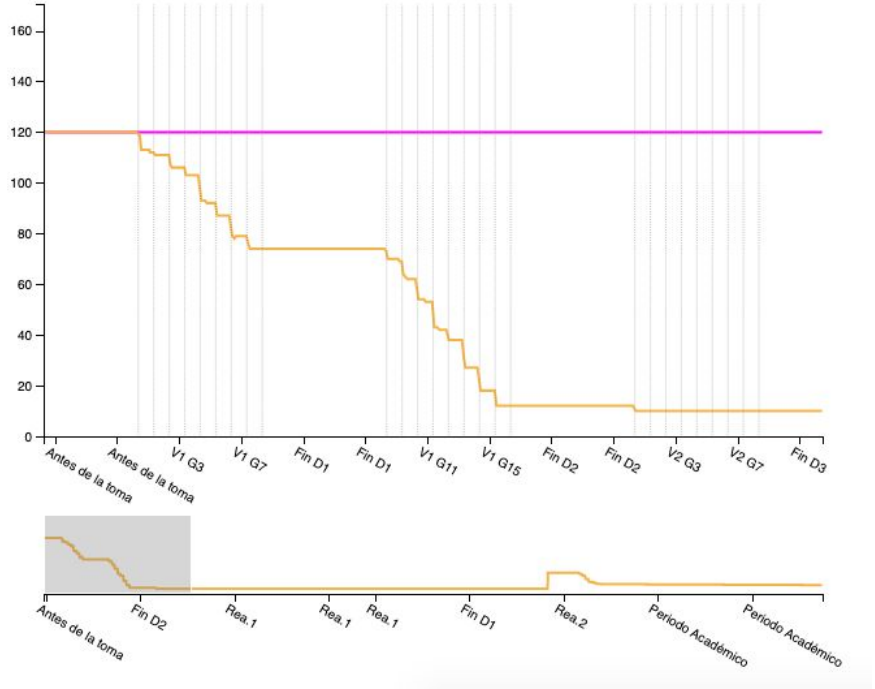


Brush en D3

¿Qué puedo hacer con Brush?

- **Una vista de minimapa y otra en detalle**





Fuente: Cupos Banner

Brush en D3

Links de interés

- d3-brush / D3 / Observable
- GitHub - d3/d3-brush: Select a one- or two-dimensional region using the mouse or touch.
- Picking, Dragging and Brushing with D3 | D3 in Depth

Agregación de datos en D3

Agregación de datos en D3

- D3 provee diferentes métodos para agrupar datos de forma eficiente.
- Algunos vistos hasta ahora:
  - `d3.min()`: El mínimo de una lista de datos.
  - `d3.max()`: El máximo de una lista de datos.
  - `d3.extent()`: El rango ([min, máx]) de una lista de datos.
- Revisaremos más métodos ahora.

Agregación de datos en D3

- `d3.mean()` → Obtener el valor promedio de una lista de datos.
- Se ocupa igual que d3.min() y d3.max()
  - `d3.mean(data)`
  - `d3.mean(data, (d) => d.sepalLength)`

Agregación de datos en D3

- `d3.group()` → Agrupar los elementos por un valor en común.
- Retorna un `InternMap` (una especie de diccionario)
- `d3.group(data, (d) => d.species)`

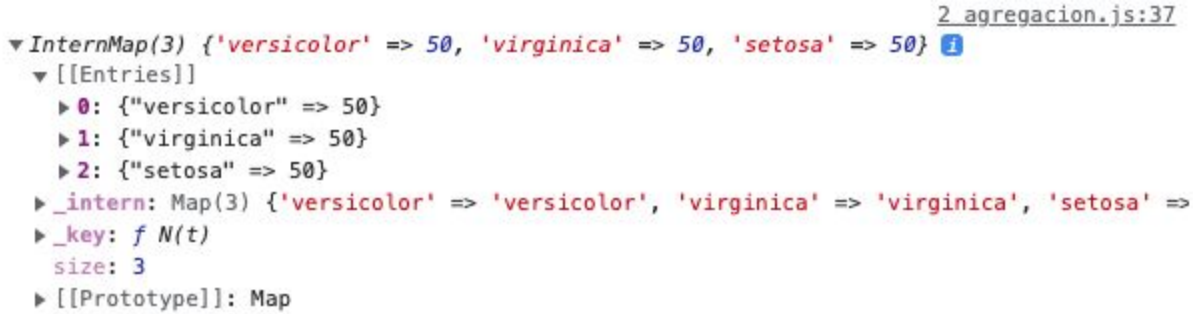
```
2 agregacion.js:26
▼ InternMap(3) {'versicolor' => Array(50), 'virginica' => Array(50), 'setosa' => Array(50)} i
  ▼ [[Entries]]
    ▶ 0: {"versicolor" => Array(50)}
    ▶ 1: {"virginica" => Array(50)}
    ▶ 2: {"setosa" => Array(50)}
  ▶ _intern: Map(3) {'versicolor' => 'versicolor', 'virginica' => 'virginica', 'setosa' =>
  ▶ _key: ƒ N(t)
    size: 3
  ▶ [[Prototype]]: Map
```

- Agregar una s al método (`d3.groups`) para retornar una lista con la información.

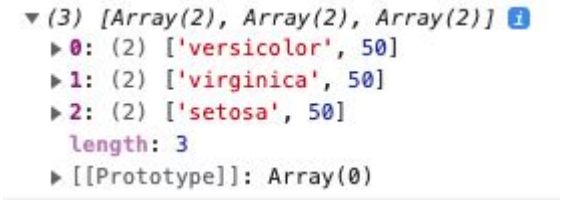
```
▼ (3) [Array(2), Array(2), Array(2)] i
  ▶ 0: (2) ['versicolor', Array(50)]
  ▶ 1: (2) ['virginica', Array(50)]
  ▶ 2: (2) ['setosa', Array(50)]
    length: 3
  ▶ [[Prototype]]: Array(0)
```

Agregación de datos en D3

- `d3.rollup()` → Agrupar los elementos por un valor en común y luego le aplica una función a cada elemento.
- `d3.rollup(data, (grupoDeDatos) => grupoDeDatos.length, (dato) => dato.species)`



- Agregar una s al método (`d3.rollups`) para retornar una lista con la información



Agregación de datos en D3

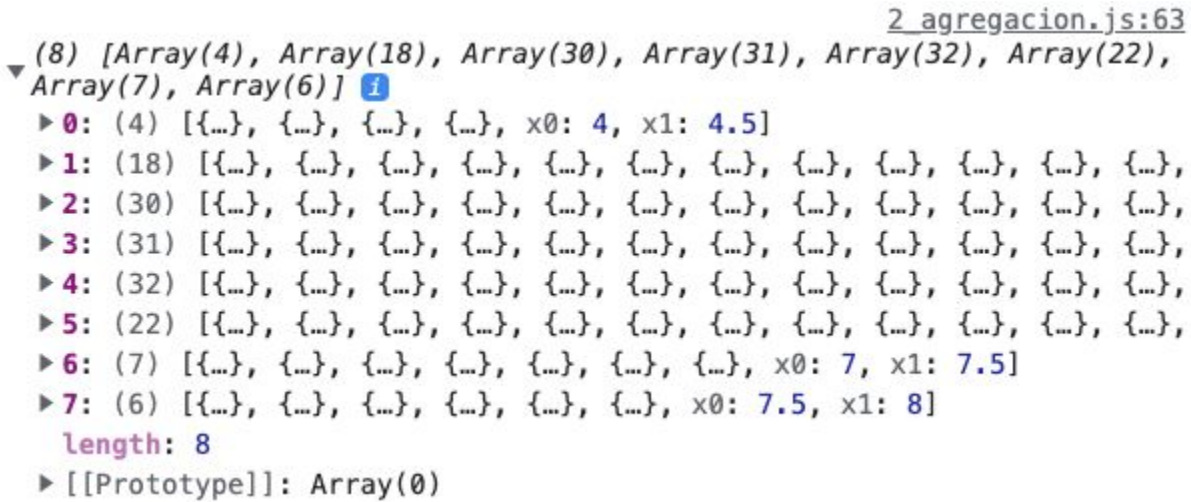
- `d3.rollup()` → Agrupar los elementos por un valor en común y luego le aplica una función a cada elemento.
- El segundo parámetro (función aplicada a cada grupo) no tiene que retornar un número si o si. Puede ser algo más complejo como un diccionaro

```
const promediosPorEspecie = d3.rollups(data,
    (grupo) => ({
      sepalLength: d3.mean(grupo, (d) => d.sepalLength),
      sepalWidth: d3.mean(grupo, (d) => d.sepalWidth),
      petalLength: d3.mean(grupo, (d) => d.petalLength),
      petalWidth: d3.mean(grupo, (d) => d.petalWidth),
    }),
    (dato) => dato.species // Agrupamos por especie
  )
```

Agregación de datos en D3

- `d3.bin()` → Agrupar los elementos en conjuntos (usualmente) de similar tamaño en función de un valor numérico.

```
d3.bin().value((d) => d.sepalLength);
console.log(binSepalLength(data));
```



Agregación de datos en D3

- `d3.bin()` → Agrupar los elementos en conjuntos (usualmente) de similar tamaño en función de un valor numérico.

```
d3.bin().thresholds(40).value((d) => d.sepalLength);
console.log(binSepalLength(data));
```

2_agregacion.js:63

```
(37) [Array(1), Array(3), Array(1), Array(4), Array(2), Array(5), Arra
y(6), Array(10), Array(9), Array(4), Array(1), Array(6), Array(7), Arr
ay(6), Array(8), Array(7), Array(3), Array(6), Array(6), Array(4), Arr
ay(9), Array(7), Array(5), Array(2), Array(8), Array(3), Array(4), Arr
ay(1), Array(1), Array(3), Array(1), Array(1), Array(0), Array(1), Arr
ay(4), Array(0), Array(1)]
```

- `thresholds` nos permite intentar que se generen esa cantidad de grupos, pero es el algoritmo de D3 que determina la cantidad final.

Agregación de datos en D3

- `d3.hexbin()` → Agrupar elementos que estén a un cierto radio en un plano cartesiano (x, y)

```
d3.hexbin()
  .radius(0.3)
  .x((d) => d.sepalLength)
  .y((d) => d.sepalWidth);

console.log(hexbin(data));

console.log(hexbin.hexagon()); // Path con la forma del hexagono
```

```
2_agregacion.js:77
(27) [Array(16), Array(17), Array(4), Array(10), Array(16), Array(2),
Array(14), Array(6), Array(2), Array(14), Array(8), Array(1), Array
▶ (4), Array(2), Array(5), Array(1), Array(4), Array(2), Array(2), Array
(3), Array(6), Array(2), Array(4), Array(1), Array(1), Array(2), Array
(1)]
```

- Ojo. Tenemos que importar esa librería
  - `<script src="https://d3js.org/d3-hexbin.v0.2.min.js"></script>`

Agregación de datos en D3

- Links de interés
  - d3.bin / D3 / Observable
  - GitHub - d3/d3-array: Array manipulation, ordering, searching, summarizing, etc.
  - GitHub - d3/d3-hexbin: Group two-dimensional points into hexagonal bins.

Hexbins en D3

Hexbins en D3

- `d3.hexbin()` → Agrupar elementos que estén a un cierto radio en un plano cartesiano (x, y)

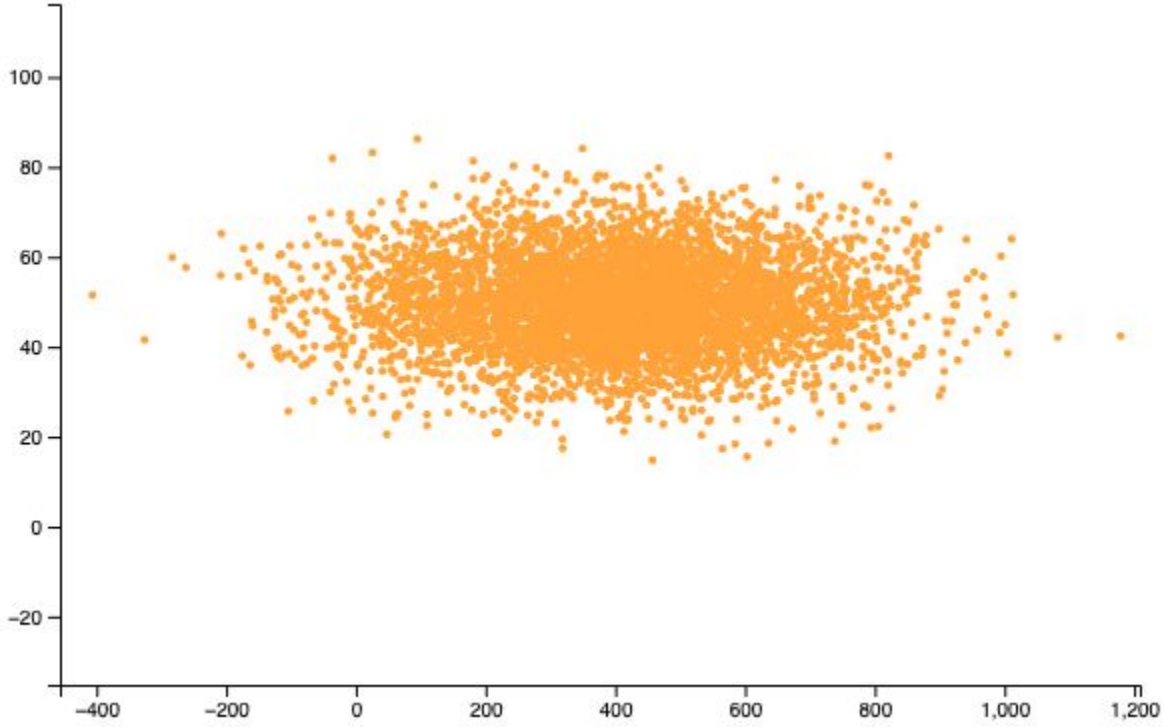
```
d3.hexbin()
  .radius(0.3)
  .x((d) => d.sepalLength)
  .y((d) => d.sepalWidth);
```

```
2_agregacion.js:77
(27) [Array(16), Array(17), Array(4), Array(10), Array(16), Array(2),
Array(14), Array(6), Array(2), Array(14), Array(8), Array(1), Array
▶ (4), Array(2), Array(5), Array(1), Array(4), Array(2), Array(2), Array
(3), Array(6), Array(2), Array(4), Array(1), Array(1), Array(2), Array
(1)]
```

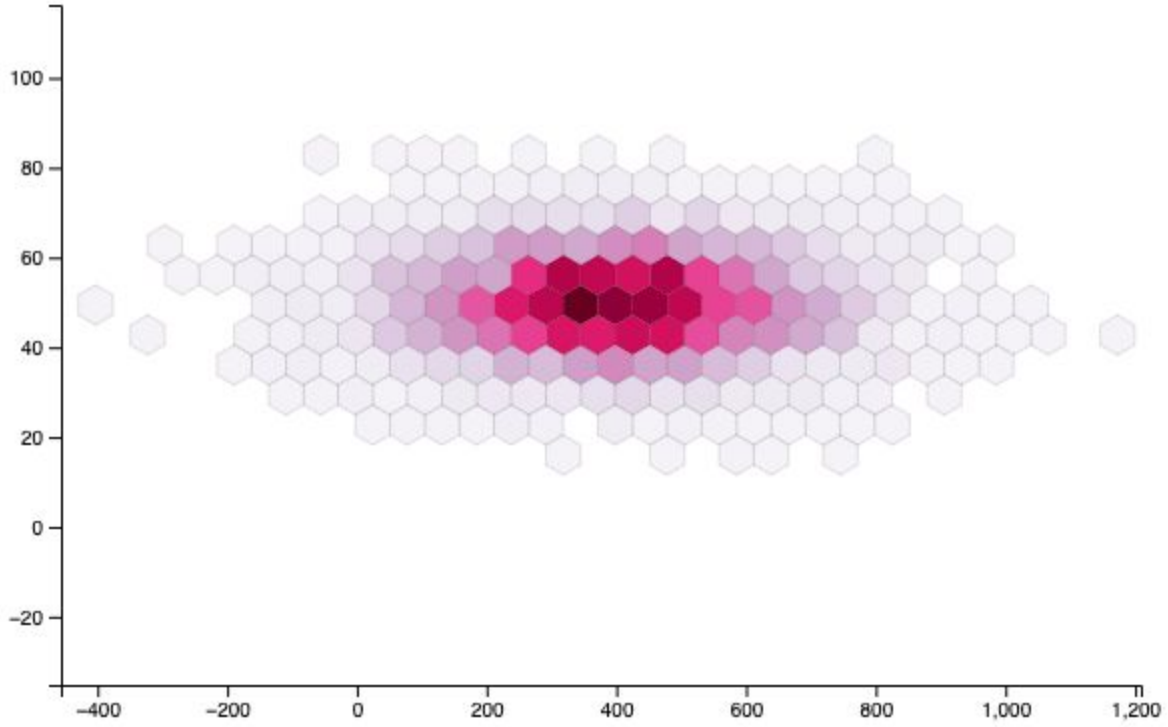
```
console.log(hexbin(data));

console.log(hexbin.hexagon()); // Path con la forma del hexagono
```

- Ojo. Tenemos que importar esa librería
  - `<script src="https://d3js.org/d3-hexbin.v0.2.min.js"></script>`

Hexbins en D3



Hexbins en D3



Hexbins en D3

Vamos al código 👨‍💻 👩‍💻

Próximos eventos

Próxima clase

- Validación de visualizaciones y privacidad de datos.

Próxima ayudantía

- **Esta semana:** no hay, es feriado.
- **La otra semana:** aplicar *brush* para crear una visualización de más alto nivel.

Tarea 4 (última tarea)

- Se publica mañana y se entrega el otro viernes.
- Criticar y mejorar visualizaciones aplicando eficiencia de canales, principios de diseño y criterios de percepción.

IIC2026
Visualización de Información

Hernán F. Valdivieso López
(2023 - 2 / Clase 19)