

# D3.JS

## 互動式資料視覺化

**Lecturer: LinJer 林哲**

evin92@gmail.com

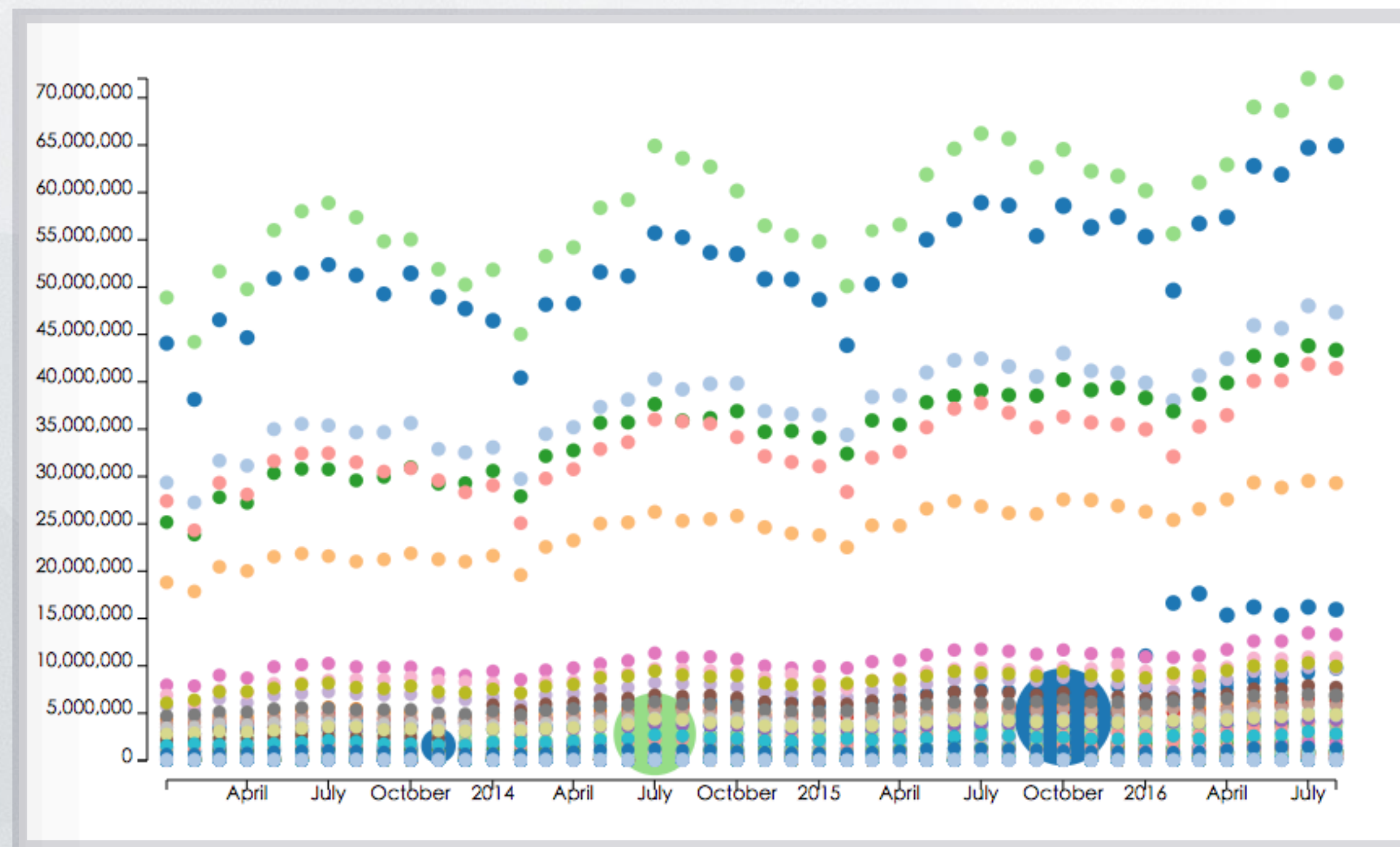




該換真實資料上場了

發票發行數量與日期  
2D散佈圖

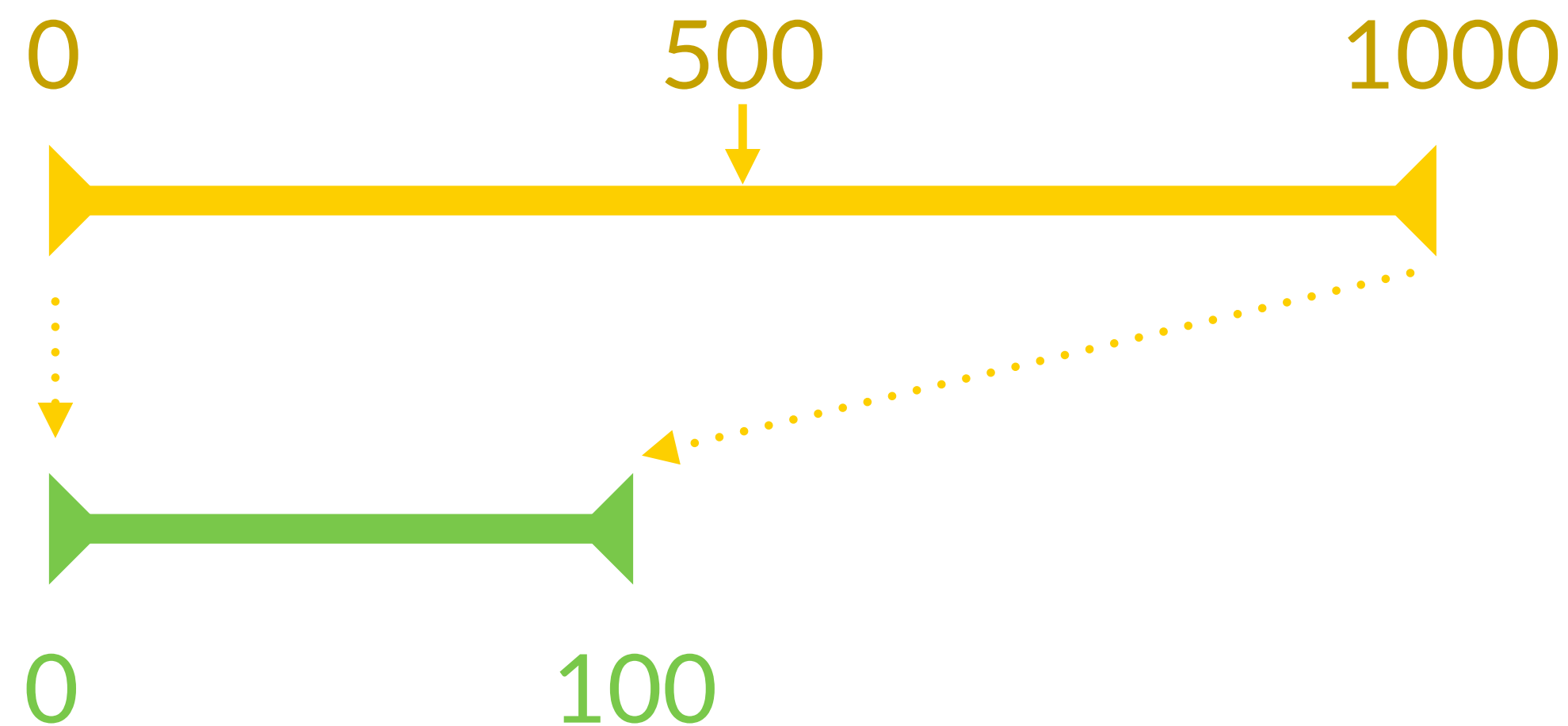
先了解: 比例尺 & 軸線





# Scale-比例尺

拿來做範圍變換-大變小



## 線性：d3.scale.linear()

```
var xScale = d3.scale.linear()  
  .domain([0, 1000])  
  .range([0, 100]);
```

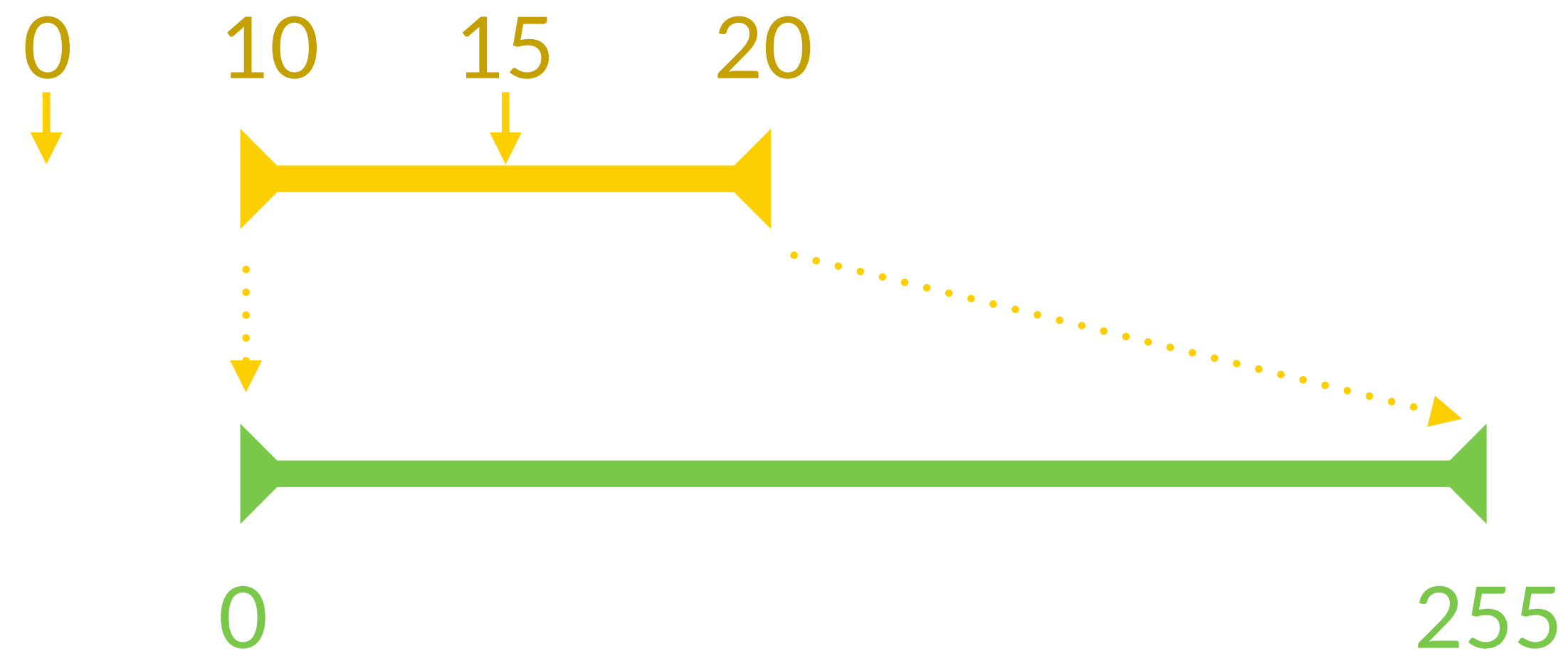
D: 輸入  
R: 輸出

在console中，用 xScale(455) 試試看

[按我連結](#)

# Scale-比例尺

拿來做範圍變換-小變大



## 線性：d3.scale.linear()

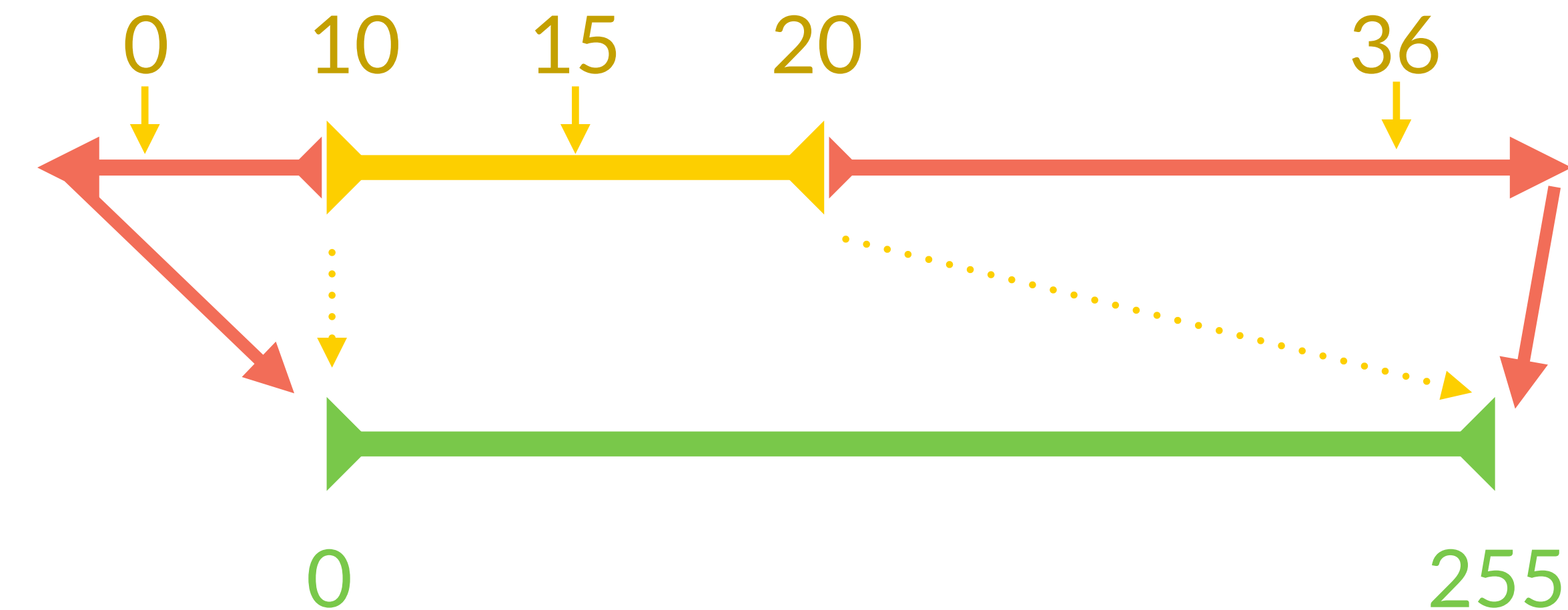
```
var xScale = d3.scale.linear()  
  .domain([10, 20])  
  .range([0, 255]);
```

D: 輸入  
R: 輸出

在console中，用 xScale(30) 試試看  
[按我連結](#)

# Scale-比例尺

clamp(true) 左右夾緊



## 線性：d3.scale.linear()

```
var xScale = d3.scale.linear()  
  .domain([10, 20])  
  .range([0, 255])  
  .clamp(true);
```

D: 輸入  
R: 輸出

在console中，用 xScale(30) 試試看  
[按我連結](#)



# 動手時間

修改random(n,m)亂數函式  
套用d3.scale.linear()

```
function random(n, m){  
    var rScale = d3.scale.linear()  
        .domain([??])  
        .rangeRound([??]);  
    return rScale(Math.random());  
}
```



## 解答時間


```
function random(n, m){  
    var rScale = d3.scale.linear()  
        .domain([0, 1])  
        .rangeRound([n, m]);  
    return rScale(Math.random());  
}
```

如果不知其原始資料大小範圍該怎麼辦？



# 比例尺的好朋友: min, max

```
var arr = [40, 10, 98];
```



## d3.min(), d3.max()

```
var arr = [40, 10, 98];
```

```
var xScale = d3.scale.linear()  
  .domain([d3.min(arr), d3.max(arr)])  
  .range([0, 255]);
```



# 比例尺的好朋友: min, max

拿好朋友來使用

```
var arr = [40, 10, 98];
var dataSet = [
  {name: "Kevin", tall: 180, age: 32},
  {name: "Helen", tall: 166, age: 15},
  {name: "Gary", tall: 172, age: 27}
];
```

## d3.max, d3.min

狀態	寫法	結果
得到最大值 max	d3.max(arr)	?
	d3.max(dataSet, function(d){ return d.tall; })	?
得到最小值 min	d3.min(arr)	?
	d3.min(dataSet, function(d){ return d.age; })	?



# 比例尺的好朋友: min, max

拿好朋友來使用

```
var arr = [40, 10, 98];
var dataSet = [
  {name: "Kevin", tall: 180, age: 32},
  {name: "Helen", tall: 166, age: 15},
  {name: "Gary", tall: 172, age: 27}
];
```

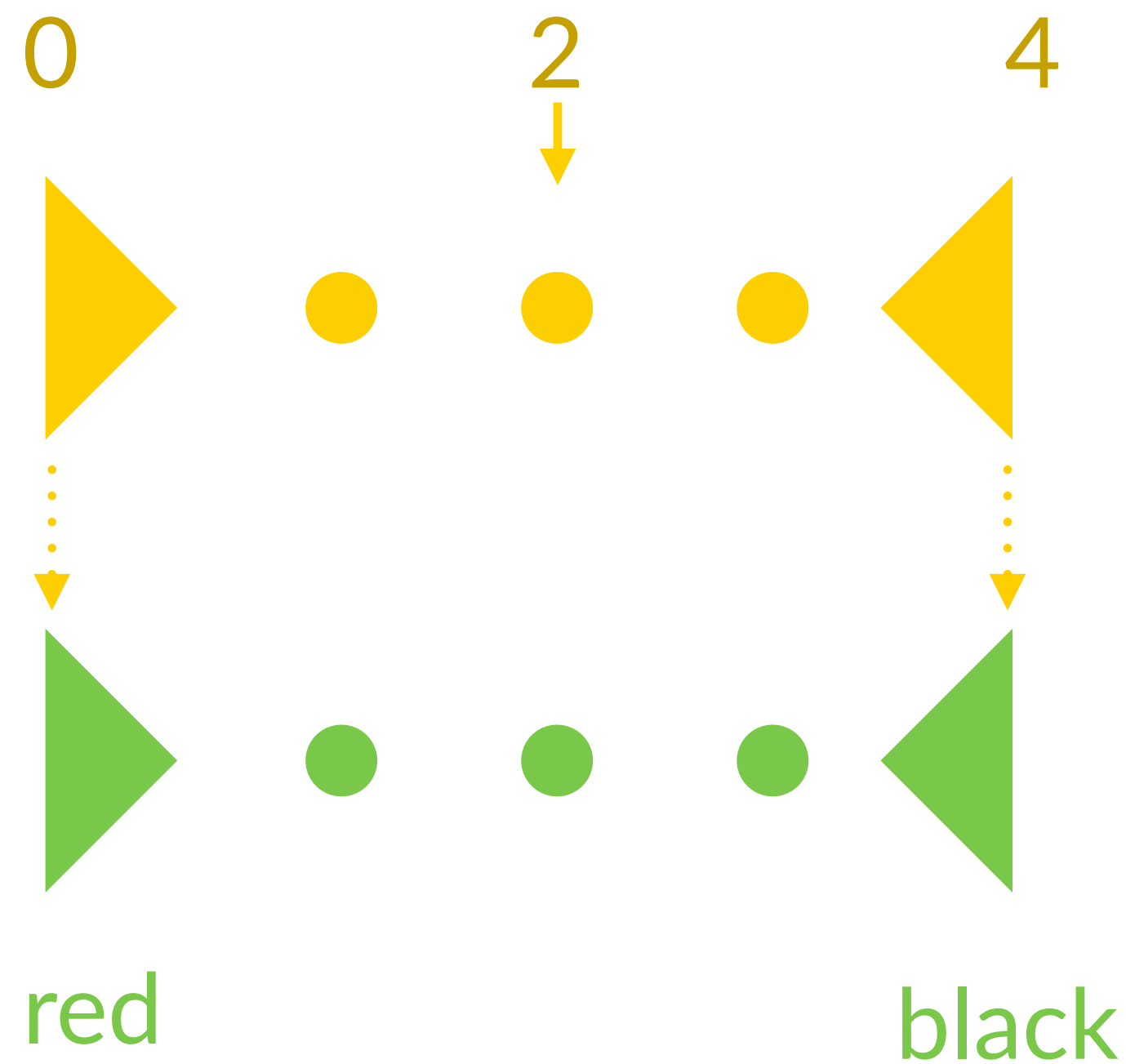
## d3.max, d3.min

狀態	寫法	結果
得到最大值 max	d3.max(arr)	98
	d3.max(dataSet, function(d){ return d.tall; })	180
得到最小值 min	d3.min(arr)	10
	d3.min(dataSet, function(d){ return d.age; })	15



# Scale-比例尺

拿來做範圍變換-不連續對應



## 序數：d3.scale.ordinal()

```
var index = [0, 1, 2, 3, 4];  
var color = ["red", "blue", "green", "yellow", "black"]
```

```
var xScale = d3.scale.ordinal()  
  .domain(index)  
  .range(color);
```

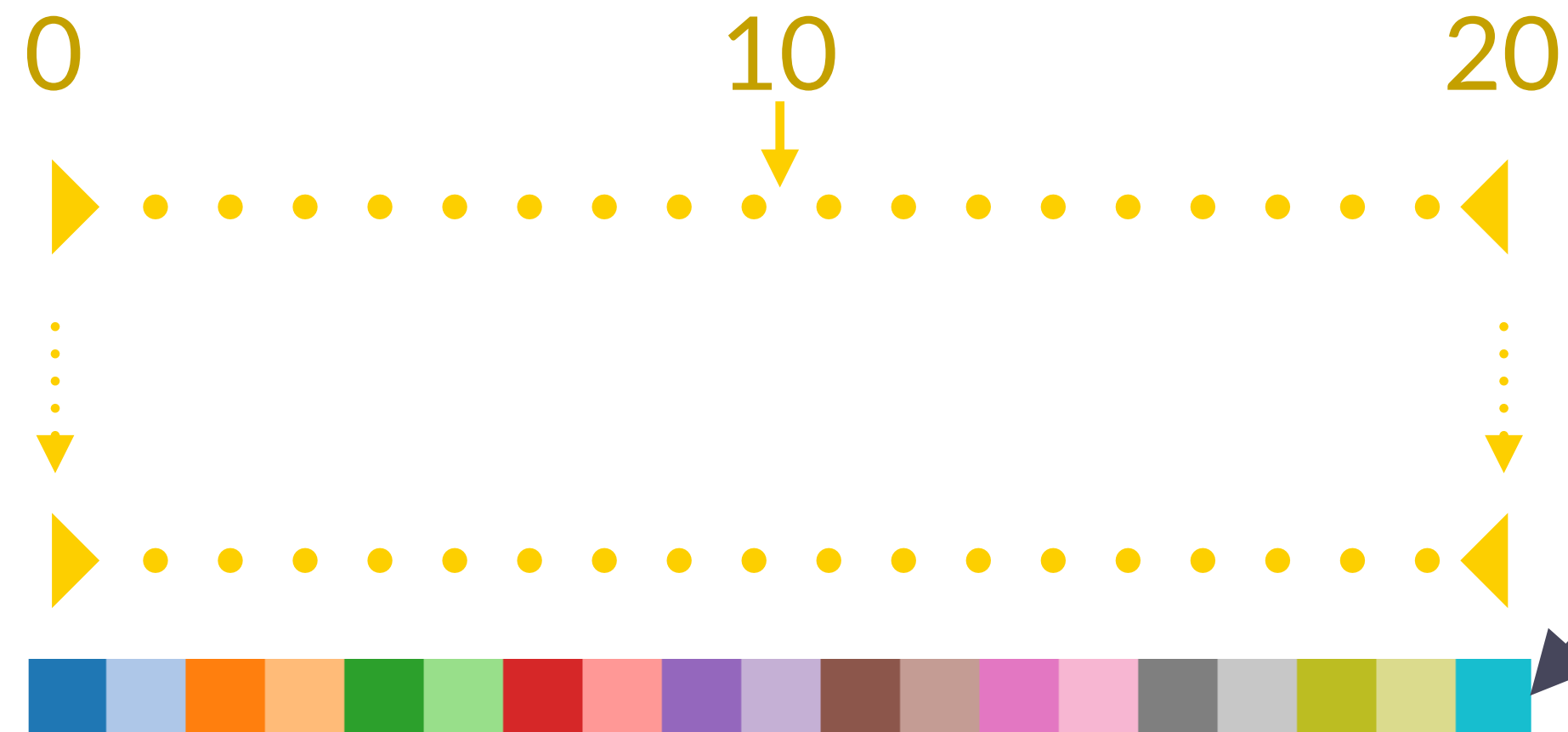
D: 輸入  
R: 輸出

在console中，用 xScale(3) 試試看  
[按我連結](#)



# Scale-比例尺

拿來做範圍變換-內建填色



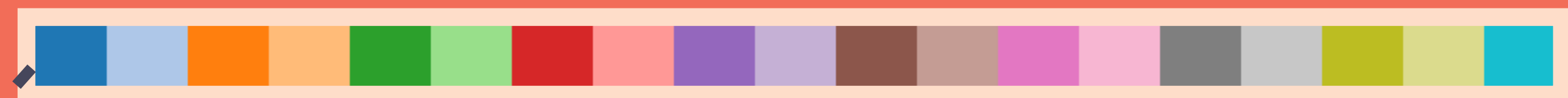
用console試試看

d3內建填色序數：  
`d3.scale.category20()`

```
var fScale = d3.scale.category10();
```



```
d3.scale.category20();
```



```
d3.scale.category20b();
```



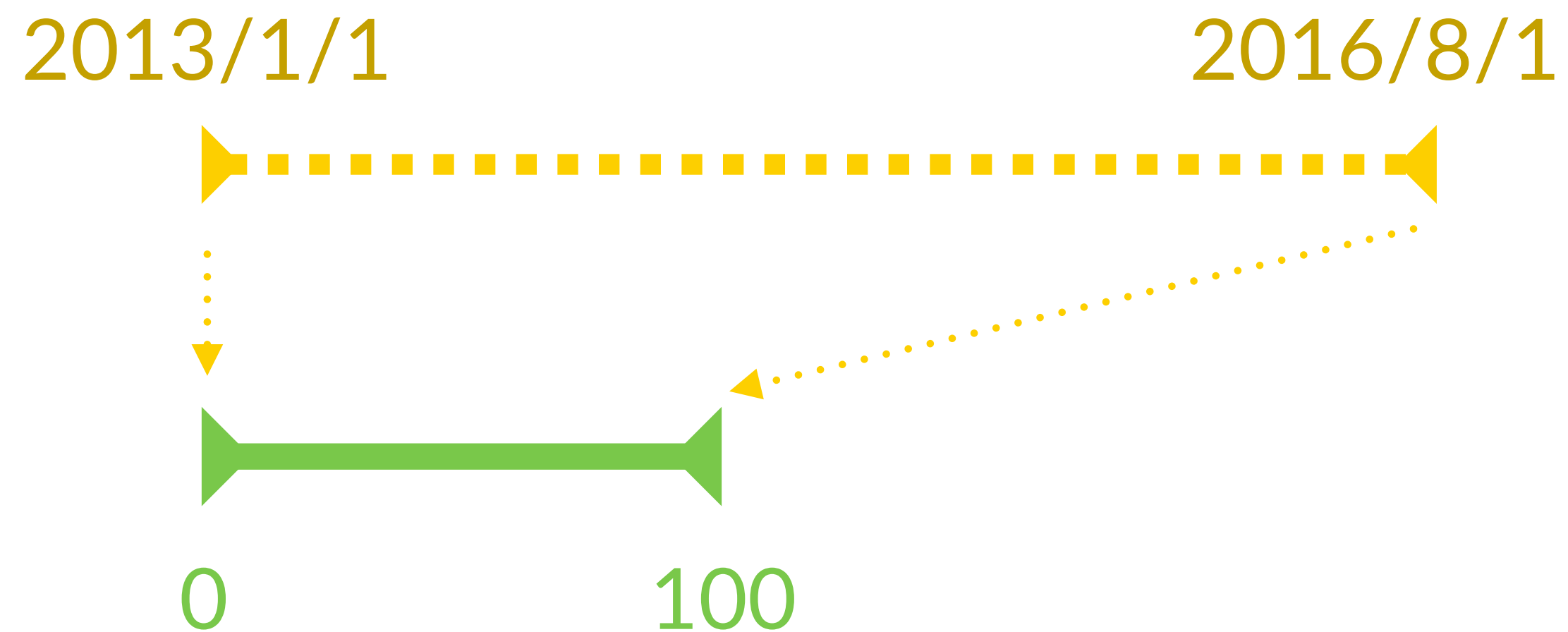
```
d3.scale.category20c();
```





# Scale-比例尺

拿來做範圍變換-時間對應



## 日期：d3.time.scale()

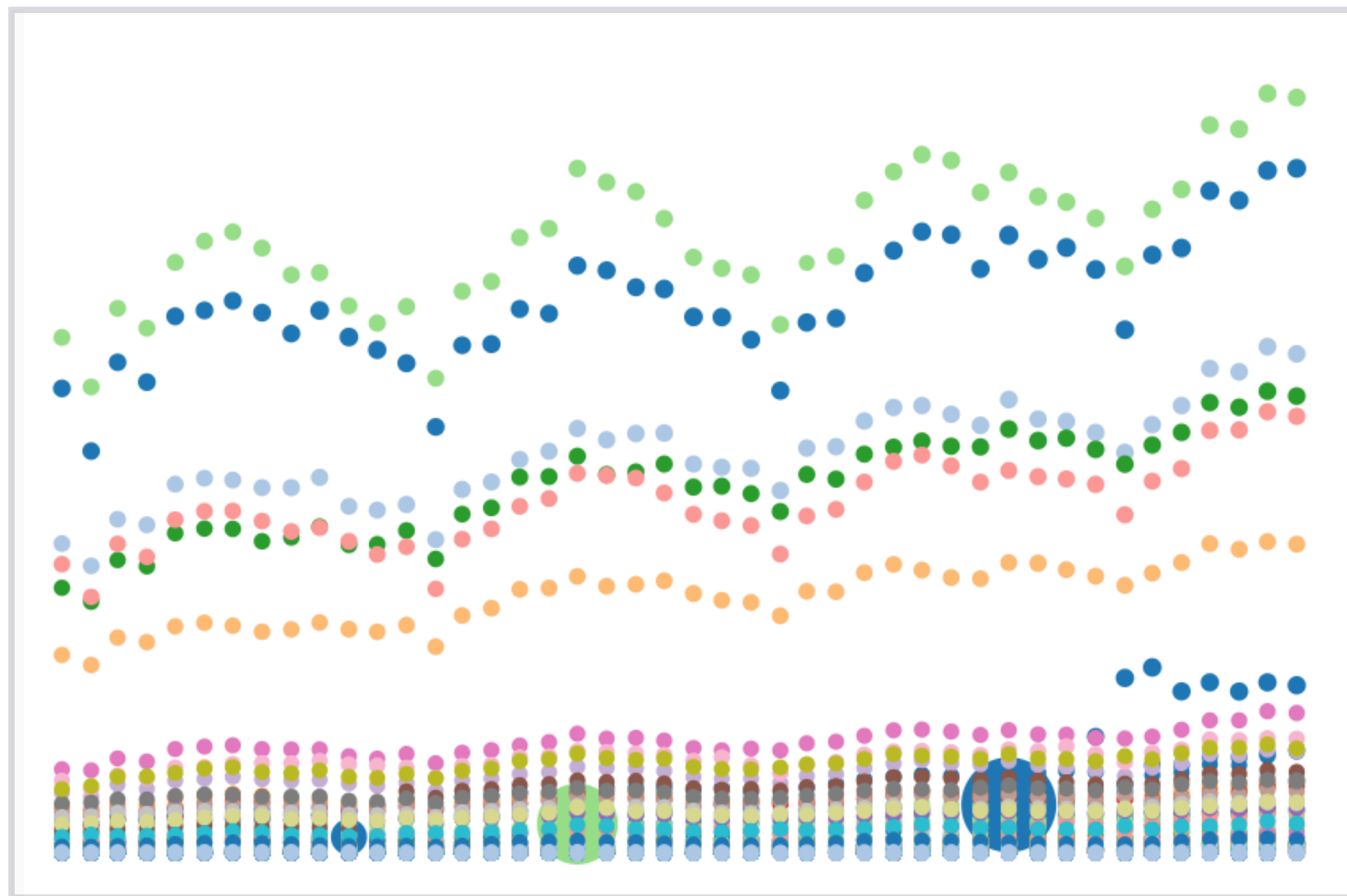
```
var xScale = d3.time.scale()  
    .domain([  
        new Date("2013-01-01"),  
        new Date("2016-08-01")  
    ])  
    .range([0, 100]);
```

D: 輸入

R: 輸出

在console中，  
用 xScale(new Date("2014-10-30")) 試試看

# 來畫畫2D散佈圖(無軸線版)



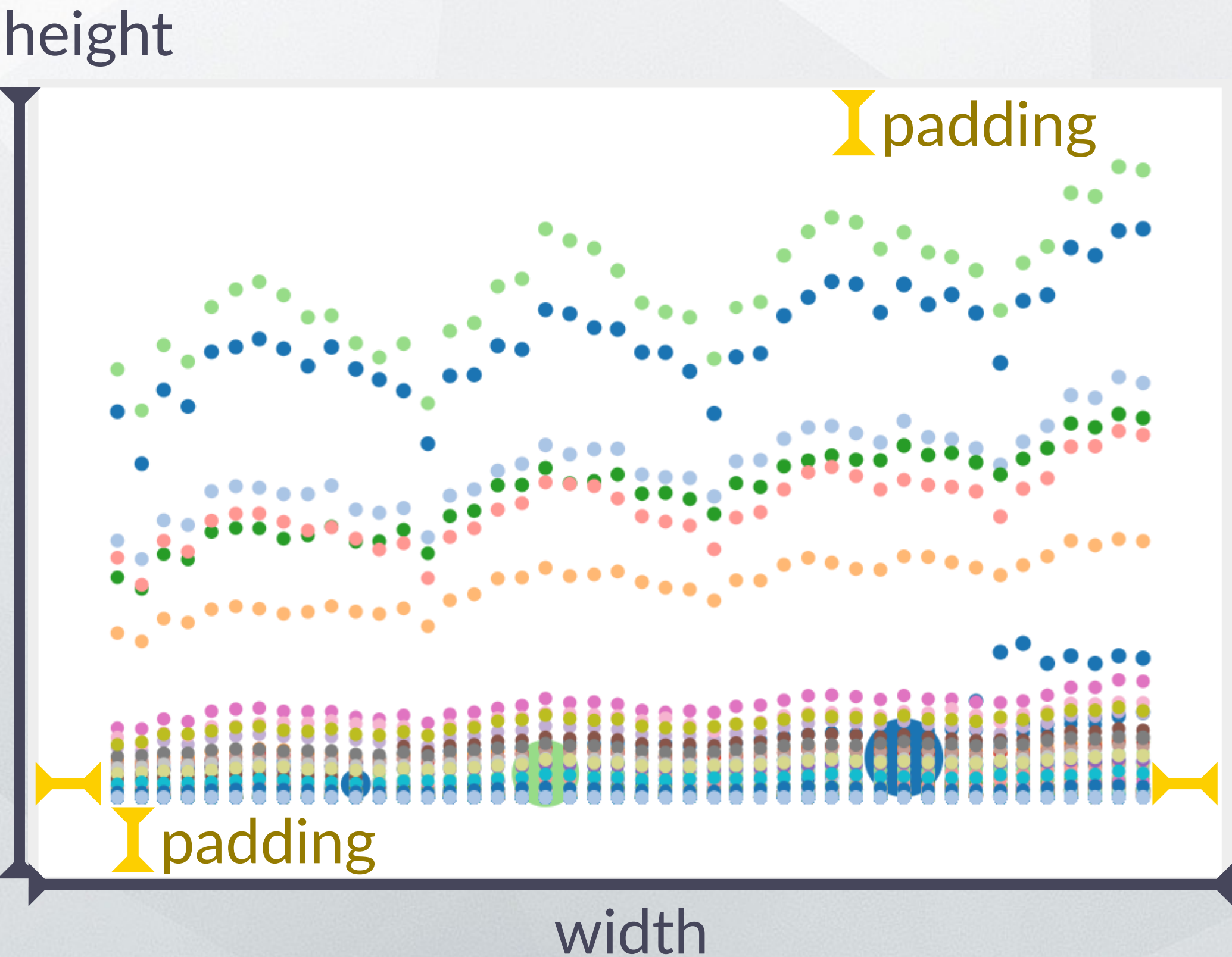
[按我下載範本](#)

建議步驟：

1. 定義width, height, padding, letterList變數
2. 建立svg()畫布環境
3. 用d3讀取csv
4. 建立bind()
5. 定義xScale, yScale, rScale, fScale比例尺(前三者range目的在決定在svg上位置)
6. 建立render()繪圖



# 定義xScale, yScale, rScale, fScale比例尺

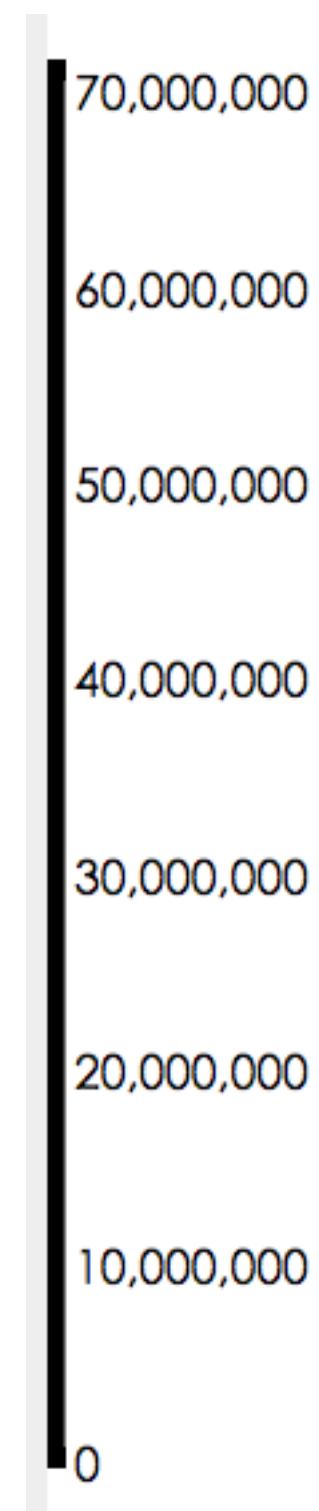


比例尺們	類別	Domain(資料)	Range(svg上位置)
xScale	日期(date)	[日期起, 日期迄]	[padding, width-padding]
yScale	數量(number)	[0, 最大數量]	[height-padding, padding]
rScale	金額(amount)	[最小金額, 最大數量]	[5,30]
fScale	縣市(amount)	序號 ex: "C" -> 2	輸出為顏色，非位置



# Axes-軸線 - 五步驟

在svg畫布中出現比例尺



第1步-產生軸線:

第2步-畫在svg上:

## d3.svg.axis()

針對svg設計的

```
var xAxis = d3.svg.axis()  
    .scale(xScale)  
    .orient("right");
```

bottom(預設): 刻度在底部

top: 刻度在頂部

left: 刻度在左方

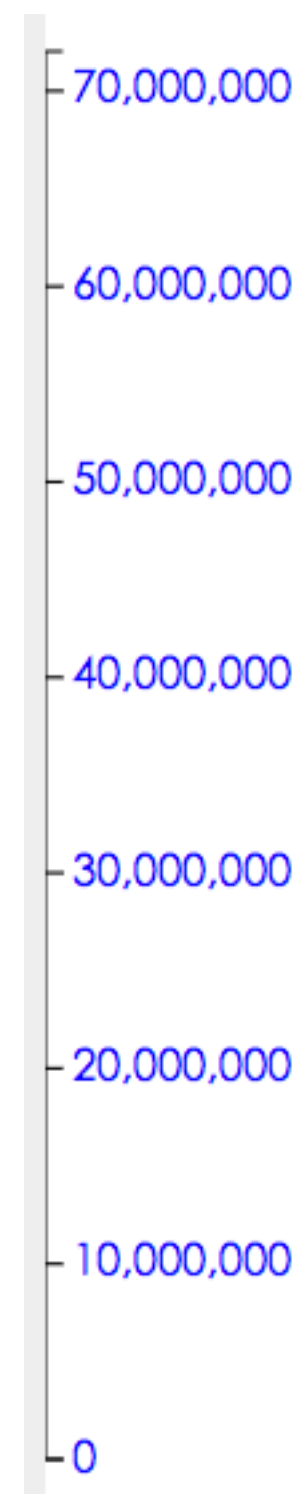
right: 刻度在右方

```
select("svg").append("g").call(xAxis);
```



# Axes-軸線 - 五步驟

在svg畫布中出現比例尺



第3步-css調整樣式:

本身由path, line, text 構成

```
.attr("class", "axis")
```

```
d3.select("svg")  
  .append("g")  
  .attr("class", "axis")  
  .call(yAxis);
```

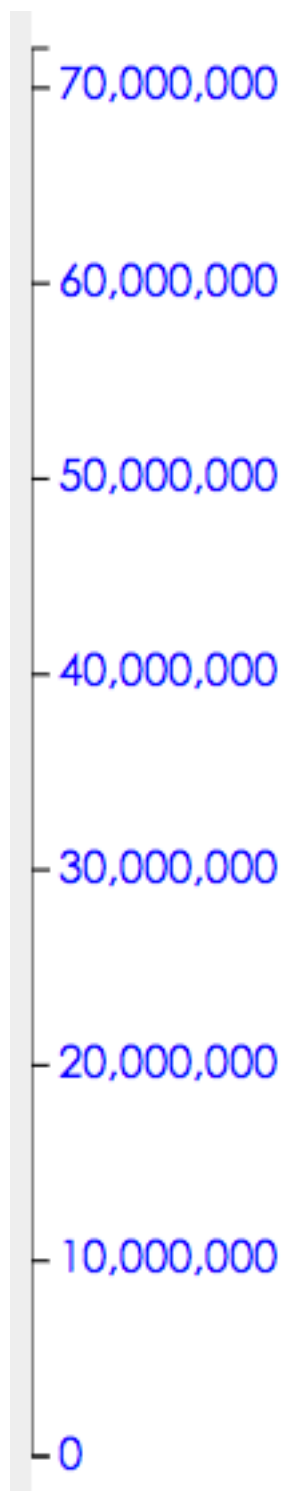
```
.axis path, .axis line{  
  fill: none;  
  stroke: black;  
  shape-rendering: auto;  
}
```

```
.axis text{  
  font-size: 11px;  
  fill: blue;  
}
```

# Axes-軸線 - 五步驟

在svg畫布中出現比例尺

第4步-刻度數量(參考值):



第5步-軸線位移:

**.ticks( )**

```
var xAxis = d3.svg.axis()  
    .scale(xScale)  
    .orient("bottom")  
    .ticks(5);
```

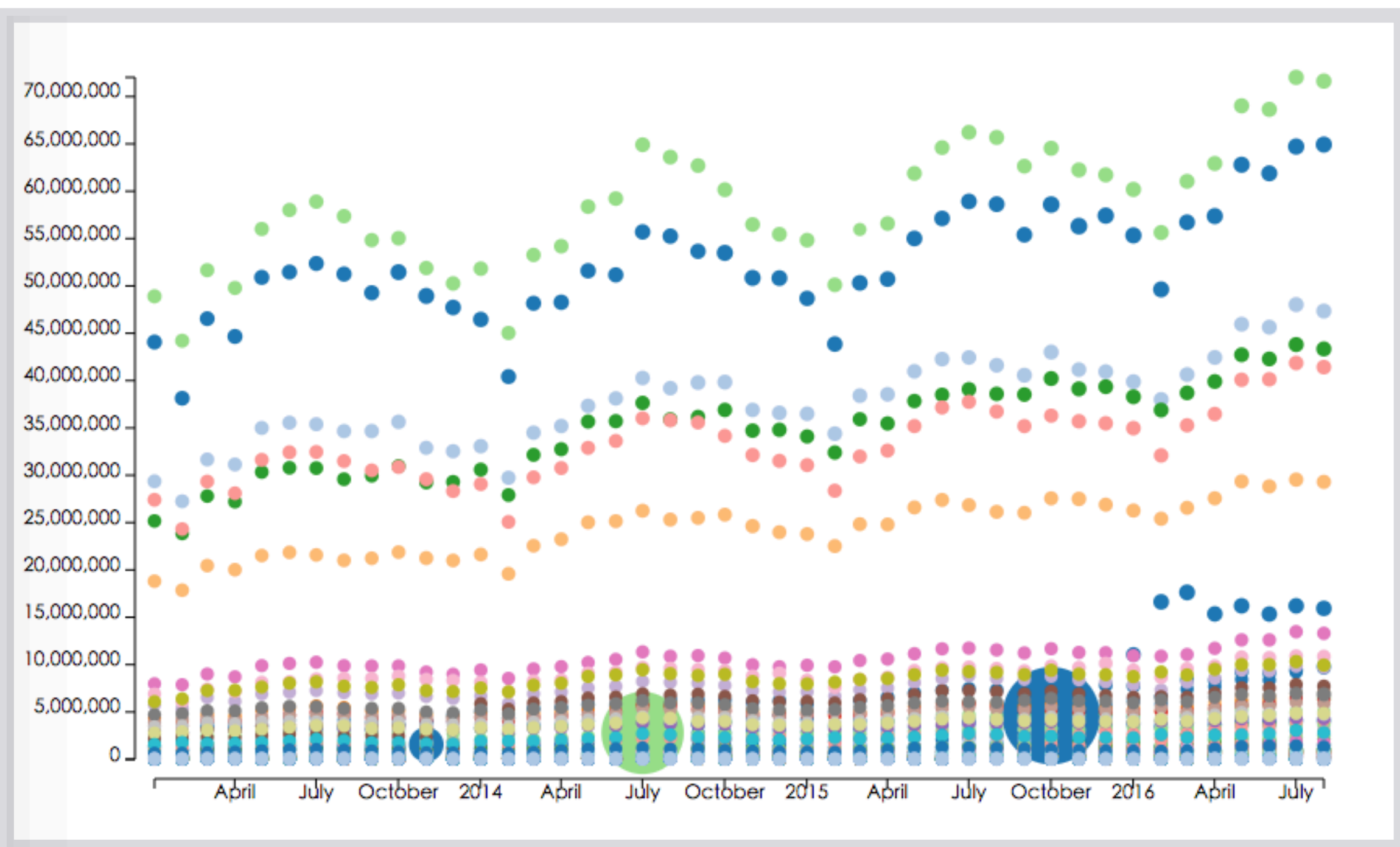
**.attr("transform", "translate("+...)**

```
d3.select("svg")  
    .append("g")  
    .attr("class", "axis")  
    .attr("transform", "translate("+x+", "+y+")")  
    .call(yAxis);
```





## 來畫畫2D散佈圖(軸線版)



建議步驟：

1. 在render()裡畫上x,y 兩軸
2. 在CSS 針對 path, line, text設定樣式
3. 調整 padding大小



```
var arr1=["A","B","A","C"];  
unique(arr1);
```

## 彙整不重複資料項

1. 建立一個unique()函式
2. 將需要彙整的資料陣列丟入unique()中

```
function unique(array){  
  var n = [];  
  去看每個array，  
  如果沒出現過就加到n中  
  return n;  
}
```

[按我開始練習](#)



## 彙整不重複資料項

傳入非單純陣列怎麼辦？

[查看unique\(\)怎麼寫](#)

```
var arr1=["A","B","A","C"];  
var arr2=[  
    {city: "台北市", cid: "A"},  
    {city: "台中市", cid: "B"},  
    {city: "台北市", cid: "A"},  
    {city: "基隆市", cid: "C"}  
];  
unique(arr2);
```



## Array.map()

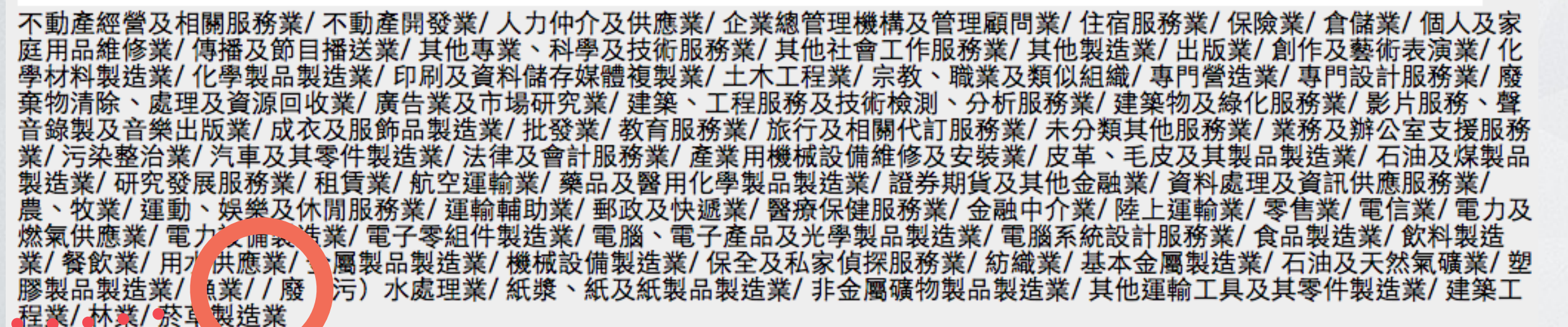
對陣列中的各元素進行操作，  
並返回一個一樣大小的新陣列

[查看執行結果](#)

```
var arr2=[  
    {city: "台北市", cid: "A"},  
    {city: "台中市", cid: "B"},  
    {city: "台北市", cid: "A"},  
    {city: "基隆市", cid: "C"}  
];  
var arr3 = arr2.map(function(d){  
    return d.cid;  
});  
unique(arr3);
```



## 使用unique()及ARRAY.map() 把所有行業別都列出來





## Array.filter()

對陣列中的各元素進行條件判斷，  
並返回一個過濾後新的陣列

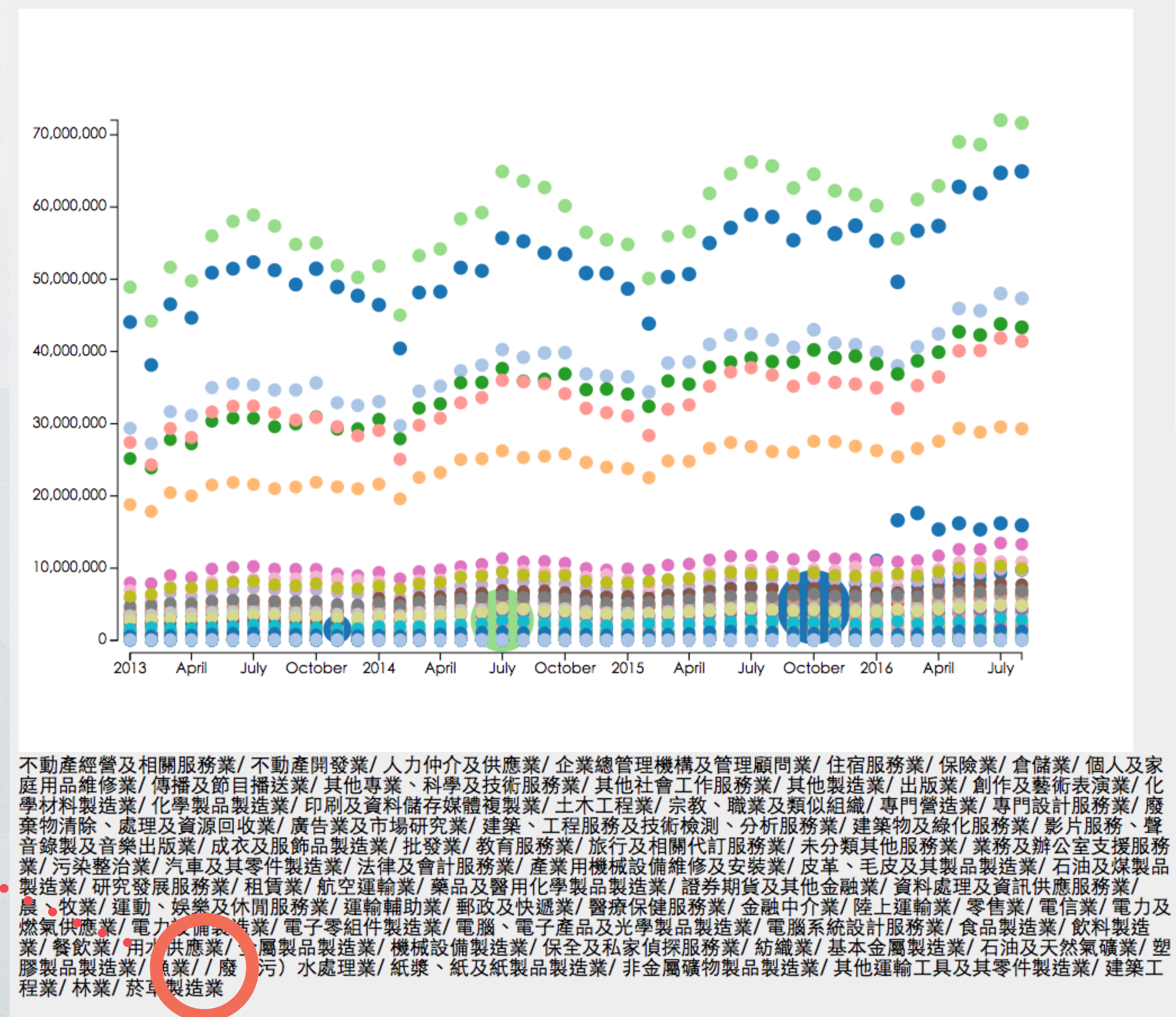
[查看執行結果](#)

```
var arr2=[
    {city: "台北市", cid: "A"},
    {city: "台中市", cid: "B"},
    {city: "台北市", cid: "A"},
    {city: "基隆市", cid: "C"}
];
var arr3 = arr2.map(function(d){
    return d.cid;
});
var arr4 = arr3.filter(function(d){
    return d !== "A";
});
unique(arr4);
```



# 動手時間

用ARRAY.filter()  
過濾無行業別的项目

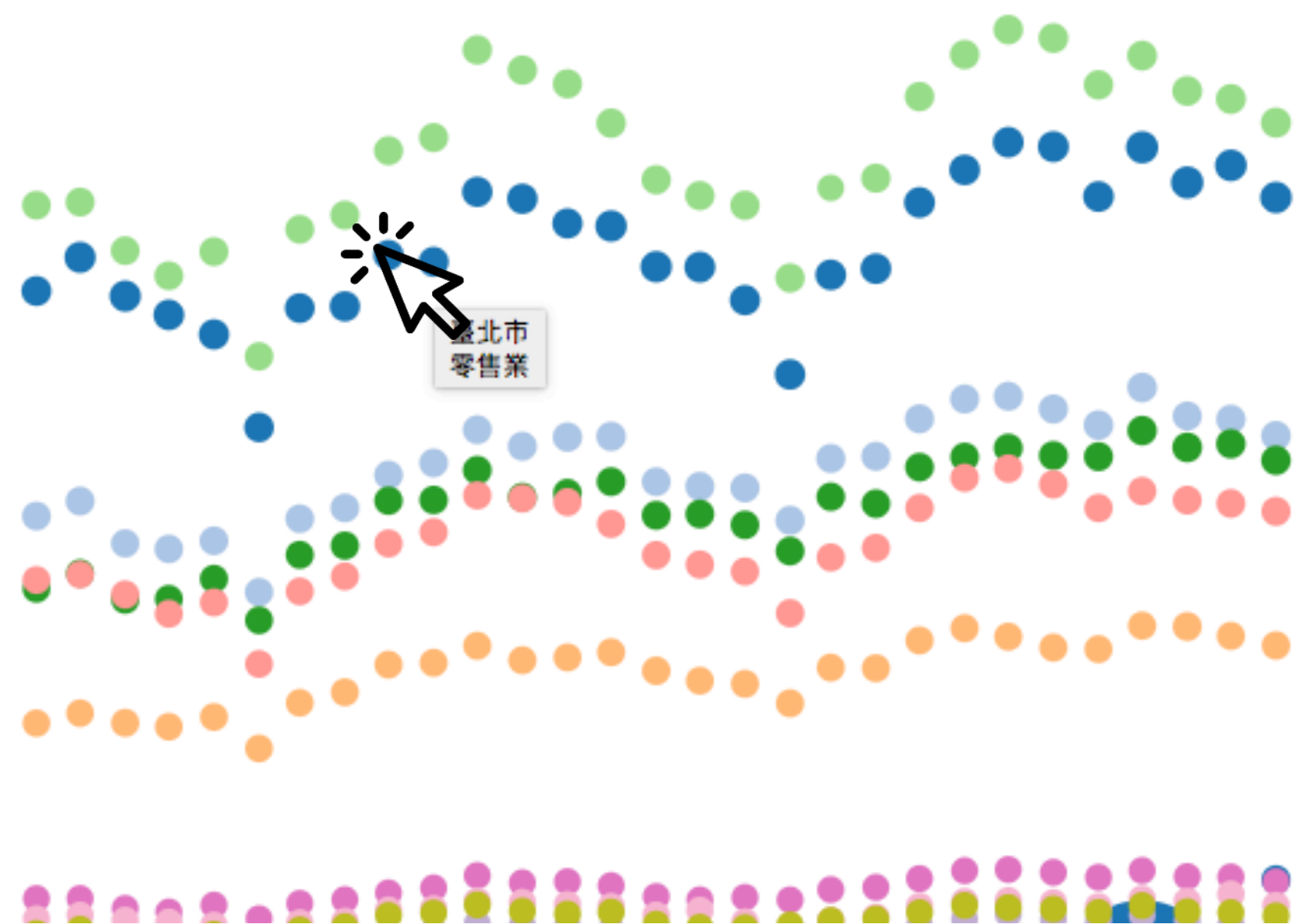






# 加入提示框 - 第一種方式

## 瀏覽器預設提示框



<title>提示內容</title>

```
d3.selectAll("circle")
  .attr({
    cx: ....
  })
  .append("title").text(function(d){
    return d.city+"\r\n"+d.industry;
  });
```