

SNU BIG DATA ACADEMY

D3.js Tutorial

조재민

Human-Computer Interaction Lab

Dept. of Computer Science & Engineering

Seoul National University

Find me on github (<http://github.com/e->) or
visit my blog (<http://www.jaeminjo.com>)

선행 지식

1. HTML, CSS

- 웹 기반 시각화이므로 웹 문서를 구성하는 기본 요소들에 대한 사전 지식이 있으면 좋습니다.

2. Javascript

- D3.js는 자바스크립트 라이브러리이므로 자바스크립트에 대한 선행 지식이 있으면 좋습니다.

간단하게 핵심적인 부분만 시작하기에 앞서 복습할 것입니다.

준비물

1. 실습 자료 다운로드 받기

<https://github.com/SNU-HCIL/2016-SNUBDA-Summer-Engineering>

2. 텍스트 에디터 및 웹 브라우저 준비하기

- 텍스트 에디터: Sublime, Vim, Emacs, Visual Studio, ...
- <http://www.sublimetext.com/2>
- 윈도우 기본 메모장은 인코딩에 주의
- 웹 브라우저: Chrome, Firefox, Edge, ...
- IE의 경우 로컬 스크립트 실행여부를 묻는 알람창이 뜨면 “차단된 콘텐츠 허용” 클릭

목차

1. 왜 D3.js를 사용(해야) 하는가?
2. HTML과 Javascript 기초 복습
3. D3.js의 기초
4. D3.js로 시각화 만들기
 1. Bar chart
 2. Scatterplot
 3. Network Diagram
5. Apache Spark와 D3.js 연동하기
6. 마무리



Data-Driven Documents

[illegible]

간단한 역사

- D3.js는 Stanford Vis Group의 연구 프로젝트
- Michael Bostock (<https://bost.ocks.org/mike/>)이 만들었고 관리
- InfoVis 2011에 발표
 - *Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-Driven Documents.*
- ProtoVis (2009)라는 차트 그리기 라이브러리의 후속 연구

Why D3.js?

- 풍부하고 다양한 예제
 - <https://github.com/mbostock/d3/wiki/Gallery>
- 바 차트나 라인 차트 같이 특정 시각화에 국한 된 것이 아닌 일반적인 시각화 생성 가능
- 짧고 간결한 코드로 시각화 및 인터랙션 디자인 가능
- 역동적인 커뮤니티
 - 현재 Github에서 사용자들이 54번째로 많이 Star한 라이브러리
- 웹 플랫폼: 별도의 설치가 필요 없고 쉽게 접근 가능함

그런데 차트 그리는 라이브러리는 많잖아요?

- Excel, Tableau, Highcharts, CharJS, Plotly, Pygal, ... 수많은 시각화 도구 및 라이브러리들
 - 고정된 차트 타입에 데이터를 입력함
 - API가 제공하는 범위 내에서만 커스터마이징 가능
- D3.js
 - 시각 요소들(사각형, 축, 색깔, ...)을 조립해서 시각화를 만듦
 - 일반적인 시각화 및 인터랙션 디자인 가능
- 빅 데이터 → 기존 시각화 개선 필요 → 일반적인 접근 → d3.js

Javascript 와 HTML 복습

JavaScript – 변수

- 변수 선언

```
var myName = "Tom";           //string 변수
var myAge = 28;                //number 변수
var isOdd = true;              //boolean 변수
var myObject = {x: 1, y: 2};  //JSON 변수
```

- 값 변경

```
myAge = 29;                    //29로 값 변경
myObject["x"] = 2;             //x인덱스의 값을 2로 변경
```

- 계산: +, ++, -, /, *, %, ...
- 비교: >, <=, ==, ===, ...

JavaScript - 배열

- 배열 선언

```
var names = ["Mao","Gandhi","Mandela"];           //문자열 배열
var mixed = [34, "candy", "blue", 11];             //혼합형 배열
var emptyArray = [];                               //빈 배열 선언
var twoD = [[1,1],[1,1]];                          //2차원 배열
var twoDObject = [{x: 1, y: 2}, {x: 3, y: 4}];     //JSON 배열
```

- 값 접근 및 변경

```
mixed[1] = "sugar";                                //"candy"였던 값을 "sugar"로 변경
twoDObject[0].x = 2;                               //1이었던 값을 2로 변경
```

JavaScript – 조건문 및 반복문

• 조건문

```
if(age < 8){  
    // do something  
}  
  
else if(age < 20){  
    // do something  
}  
  
else{  
    // do something  
}
```

• 반복문

```
for(초기화; 반복조건; 값변경){  
    // do something  
}
```

예)

```
for(var i = 1; i < 6; i++){  
    console.log(i);  
}
```



1
2
3
4
5

<콘솔창>

JavaScript – 함수

- 함수 정의

```
var divideByThree = function(number){ //3으로 나눈 후 출력하는 함수
    var val = number / 3;
    console.log(val);
};
divideByThree(6); //함수 호출, "2" 출력
```

- 같은 함수, 다양한 표현

- ```
function divideByThree(number){
 ...
};
```
- ```
function divideByThree(number){ ... };
```

JavaScript – 함수 (Cont'd)

- 값을 Return하는 함수

```
var timesTwo = function(number){  
    return number * 2;  
};  
var newNumber = timesTwo(12); //변수에 24 대입
```

- 여러개의 매개변수를 가지는 함수

```
var areaBox = function(length, width) {  
    return length * width;  
};  
var area = areaBox(10, 20);
```

JavaScript – 전역 변수와 지역 변수

- 전역 변수

```
var my_number = 12;  
var timesTwo = function(number){  
    my_number = number * 2;  
};
```

- 전역 변수 vs 지역 변수

```
var my_number = 12;  
var timesTwo = function(number){  
    var my_number;  
    my_number = number * 2; //지역 변수 my_number  
};
```

HTML & CSS

- 기본 구조

```
<html>
  <head>
    <title>My Webpage</title>
  </head>

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

- 계층적 구조를 가지는 태그들로 구성

열고 닫는 태그 둘이 한쌍을 이룸 (e.g., <html> ... </html>)

HTML & CSS (Cont'd)

- 태그안에 style을 지정하거나 attribute를 변경함으로써 HTML 요소들의 속성과 외형을 정의할 수 있음
- CSS를 이용하면 HTML 요소들의 Style을 일괄적으로 정의할 수 있음

- Attribute 설정

```

```

- Style 설정

```
<p>paragraph</p>
```



```
<p style="font-size: 14px; color: orange;  
font-family: Bodoni">paragraph</p>
```

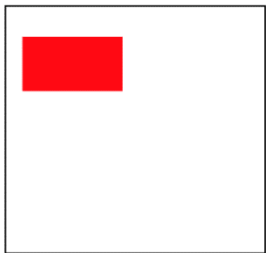
A truly spectacular paragraph!



A truly spectacular paragraph!

HTML & CSS (Cont'd)

- SVG (Scalable Vector Graphics)
 - 텍스트 기반의 벡터 이미지 포맷
 - 대부분의 현대 웹 브라우저들이 SVG를 지원
 - 예) Chrome, Safari, Firefox, IE, ...
 - 기본적인 시각적 요소들 제공
 - 예) 사각형 <rect>, 원 <circle>, 타원 <ellipse>, Path <path>, ...



```
<html><body>  
  <svg width="150" height="150">  
    <rect x="10" y="20" width="60" height="30" fill=red/>  
  </svg>  
</body></html>
```

Basic SVG Elements (geometric attributes)



rect (x,y,width,height)



circle (cx, cy, r)



ellipse (cx, cy, rx, ry)



line (x1, y1, x2, y2)



path (d)



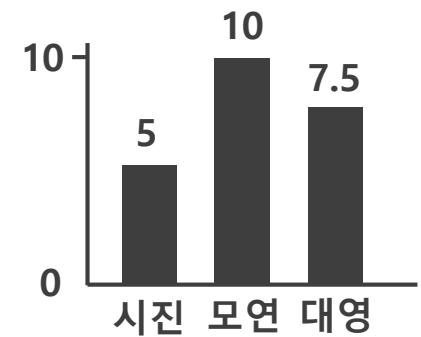
polygon (points)

D3.js 기초

Binding Data to Visualization

이름	값
시진	5
모연	10
대영	7.5

Table

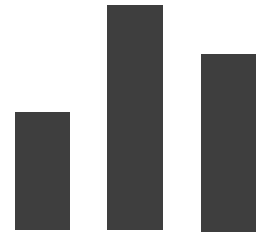


Visualization

Binding Data to Visualization (Cont'd)

이름	값
시진	5
모연	10
대영	7.5

Table



Visualization

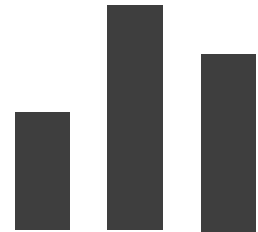
Binding Data to Visualization (Cont'd)

이름	값
시진	5
모연	10
대영	7.5

Table

```
[  
  {"이름": "시진",  
   "값": 5},  
  {"이름": "모연",  
   "값": 10},  
  {"이름": "대영",  
   "값": 7.5}  
]
```

JSON



Visualization

Binding Data to Visualization (Cont'd)

이름	값
시진	5
모연	10
대영	7.5

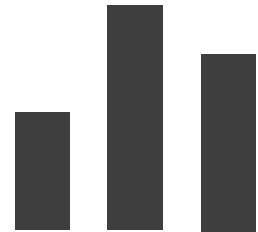
Table

```
[  
  {"이름": "시진",  
   "값": 5},  
  {"이름": "모연",  
   "값": 10},  
  {"이름": "대영",  
   "값": 7.5}  
]
```

JSON

```
<svg>  
  <rect ...></rect>  
  <rect ...></rect>  
  <rect ...></rect>  
</svg>
```

SVG



Visualization

Binding Data to Visualization (Cont'd)

이름	값
시진	5
모연	10
대영	7.5

Table

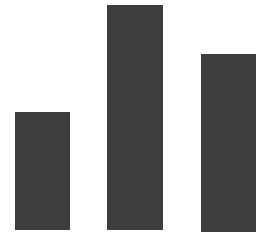
```
[  
  {"이름": "시진",  
   "값": 5},  
  {"이름": "모연",  
   "값": 10},  
  {"이름": "대영",  
   "값": 7.5}  
]
```

JSON

D3.js

```
<svg>  
<rect ...></rect>  
<rect ...></rect>  
<rect ...></rect>  
</svg>
```

SVG



Visualization

데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```

화면 상에는 그대로

데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]  
  
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]  
  
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
</svg>
```

- Update
- Enter
- Exit

데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5}  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]  
  
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]  
  
<svg>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

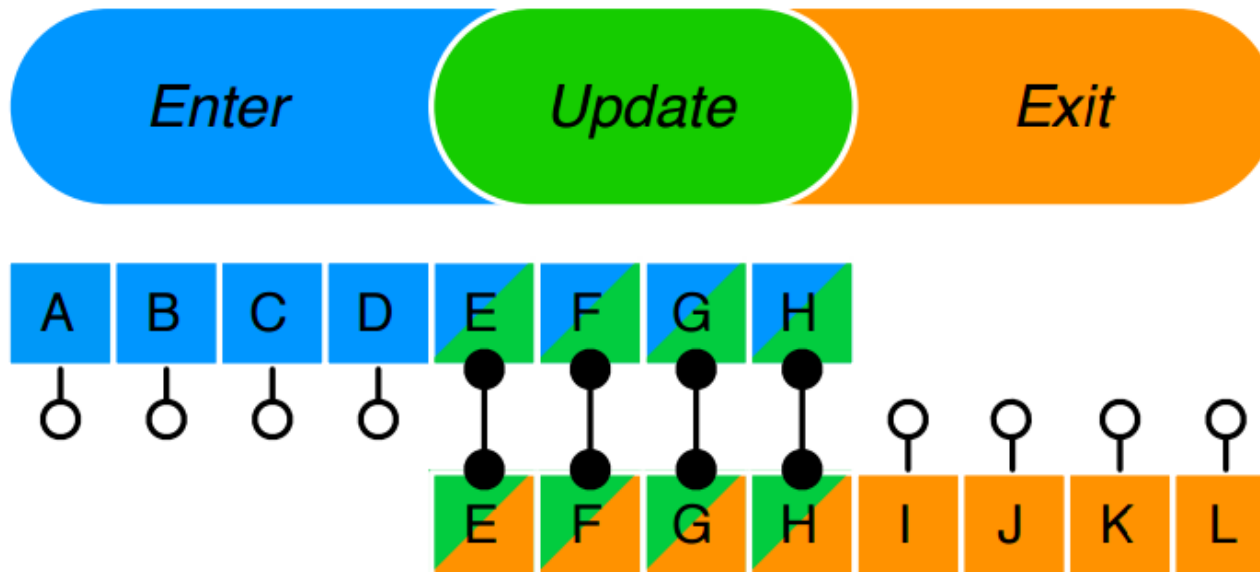
화면상의 차트와 데이터가 일치

데이터와 노드간의 조인 (join)

자바스크립트 상의
데이터 (JSON)HTML (SVG) 상의
노드

Data

Nodes



기본적인 API

- `d3.select(selector) => selection`
 - `selector`에 해당하는 요소들을 하나 선택한 셀렉션을 반환
 - `d3.select('body') => body` 태그를 선택
- `d3.selectAll(selector) => selection`
 - `selector`에 해당하는 요소들을 모두 선택한 셀렉션을 반환
 - `d3.selectAll('rect') => 페이지내 모든 rect`를 선택
 - `d3.select('svg').selectAll('rect') => svg 태그내에 있는 모든 rect`를 선택
- `selection.append(tag) => selection`
 - 셀렉션에 자식으로 `tag`를 추가
 - `d3.select('body').append('div')`

기본적인 API

- `selection.attr(key, value) => selection`
 - selection에 선택된 요소들의 속성을 변경
 - `<rect height="200"></rect>`
 - `d3.select('rect').attr('height', '300')`
 - `<rect height="300"></rect>`
- `selection.text(text) => selection`
 - selection에 선택된 요소들의 내용을 변경
 - `<text>hello</text>`
 - `d3.select('text').text('world')`
 - `<text>world</text>`

기본적인 API

- selection.data(*array*)
 - selection에 선택된 요소들과 *array*를 조인 (join)
 - 기본적으로 update 셀렉션을 가져옴
- selection.data(*array*).enter()
 - enter 셀렉션을 가져옴
 - selection.data(array).enter().append('rect')
- selection.data(*array*).exit()
 - exit 셀렉션을 가져옴
 - selection.data(array).exit().remove()

D3.js로 시각화 만들기

Bar chart 만들기 (1)

1. 1_barchart 디렉토리 안에 있는 barchart_skeleton.js를 텍스트 에디터로 열기
 2. barchart_skeleton.html을 브라우저에서 열기
 3. js 파일을 수정한 후 저장하고 브라우저를 새로고침하여 결과를 확인
- 모든 html파일은 실습 내내 수정하실 필요가 없습니다.
 - 완성된 코드는 barchart.js에 있습니다.

Bar chart 만들기 (2)

```
var data = [  
  {"name": "Sally", "value": 5},  
  {"name": "Tom", "value": 10},  
  {"name": "John", "value": 7.5}  
]
```

```
<svg>
```

```
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략

Javascript

HTML

Bar chart 만들기 (3)

```
var data = ...
```

```
var svg = d3.select('svg')
```

```
<svg>
```

```
</svg>
```

Javascript

HTML

Bar chart 만들기 (4)

```
var data = ...
```

```
<svg>
```

```
</svg>
```

```
var svg = d3.select('svg')
```

```
svg
```

- svg 내부에는 <rect>가 하나도 없으므로 빈 선택션을 돌려줌

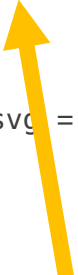
```
.selectAll('rect')
```

Javascript

HTML

Bar chart 만들기 (5)

```
var data = ...  
  
var svg = d3.select('svg')  
  
svg  
  .selectAll('rect') // 빈 선택션  
  .data(data)
```



```
<svg>  
  
</svg>
```

Javascript

HTML

Bar chart 만들기 (6)

```
var data = ...
```

```
<svg>
```

```
</svg>
```

```
var svg = d3.select('svg')
```

```
svg
```

```
  .selectAll('rect')
```

```
  .data(data)
```

```
  .enter()
```

Javascript

HTML

Bar chart 만들기 (7)

```
var data = ...  
  
var svg = d3.select('svg')  
  
svg  
  .selectAll('rect')  
  .data(data)  
  .enter()
```

<svg>	•
{ "name": "Sally", "value": 5 }	•
{ "name": "Tom", "value": 10 }	•
{ "name": "John", "value": 7.5 }	•
</svg>	

Javascript

HTML

Bar chart 만들기 (8)

```
var data = ...
```

```
var svg = d3.select('svg')
```

```
svg
```

```
  .selectAll('rect')
```

```
  .data(data)
```

```
  .enter()
```

```
    .append('rect')
```

Javascript

```
<svg>
```

```
  <rect></rect>
```

```
  <rect></rect>
```

```
  <rect></rect>
```

```
</svg>
```

자바스크립트로 HTML 요소를 생성
(코드 입력 X)

HTML

Bar chart 만들기 (9)

```
var data = ...
```

```
var svg = d3.select('svg')    {"name": "Sally", "value": 5}
```

```
    {"name": "Tom", "value": 10}
```

```
svg
```

```
    {"name": "John", "value": 7.5}
```

```
  .selectAll('rect')
```

```
  .data(data)
```

```
  .enter()
```

```
    .append('rect')
```

```
    .attr('width', 100)
```

```
    .attr('height', 20)
```

```
    .style('fill', 'steelblue')
```

```
<svg>
```

```
  <rect width="100" height="20" style="fill:steelblue"></rect>
```

```
  <rect width="100" height="20" style="fill:steelblue"></rect>
```

```
  <rect width="100" height="20" style="fill:steelblue"></rect>
```

```
</svg>
```



Bar chart 만들기 (10)

```
var data = ...
var svg = d3.select('svg')
```

```
svg
    .selectAll('rect')
    .data(data)
    .enter()
    .append('rect')
    .attr('width', 100)
    .attr('height', 20)
    .style('fill', 'steelblue')
    .attr('transform', function(d, i){
        return 'translate(0,' + i * 30 + ')';
    });
```

```
<svg>
  <rect width="100" height="20"
    style="fill:steelblue"
    transform="translate(0, 0)"></rect>
  <rect width="100" height="20"
    style="fill:steelblue"
    transform="translate(0, 30)"></rect>
  <rect width="100" height="20"
    style="fill:steelblue"
    transform="translate(0, 60)"></rect>
</svg>
```



Bar chart 만들기 (11)

```

var data = ...

var svg = d3.select('svg')

svg
    .selectAll('rect')
    .data(data)
    .enter()
    .append('rect')
    .attr('width', function(d, i){
        return d.value * 100;
    })
    .attr('height', 20)
    .style('fill', 'steelblue')
    .attr('transform', function(d, i){
        return 'translate(0,' + i * 30 + ')';
    });

```

```

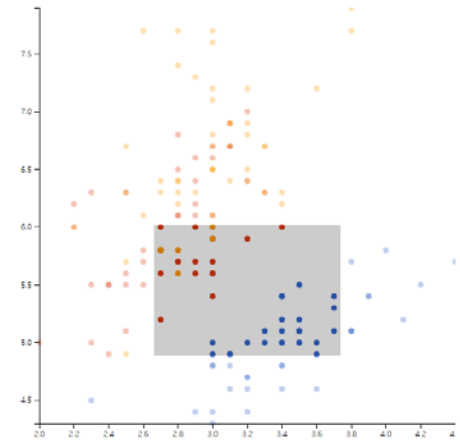
<svg>
  <rect width="50" height="20"
    style="fill:steelblue"
    transform="translate(0, 0)"></rect>
  <rect width="100" height="20"
    style="fill:steelblue"
    transform="translate(0, 30)"></rect>
  <rect width="75" height="20"
    style="fill:steelblue"
    transform="translate(0, 60)"></rect>
</svg>

```



산점도 만들기 (1)

1. 2_scatterplot 디렉토리 안에 있는 scatterplot_skeleton.js를 텍스트 에디터로 열기
 2. scatterplot_skeleton.html을 브라우저에서 열기
 3. js 파일을 수정한 후 저장하고 브라우저를 새로고침하여 결과를 확인
- 모든 html파일은 실습 내내 수정하실 필요가 없습니다.
 - 완성된 코드는 scatterplot.js에 있습니다.



MEAN(width) = 3.15, MEAN(length) = 5.42

산점도 만들기 (2)

- 사용할 데이터: data/iris.tsv
- 붓꽃의 종류별 꽃잎과 꽃받침의 길이와 너비가 탭으로 구분되어있음
 - setosa, versicolor, virginica
- [꽃받침 길이] [꽃받침 너비] [꽃잎 길이] [꽃잎 너비] [종류]
- 꽃받침의 길이와 너비로 산점도를 그리고 이를 개선해 봅시다.

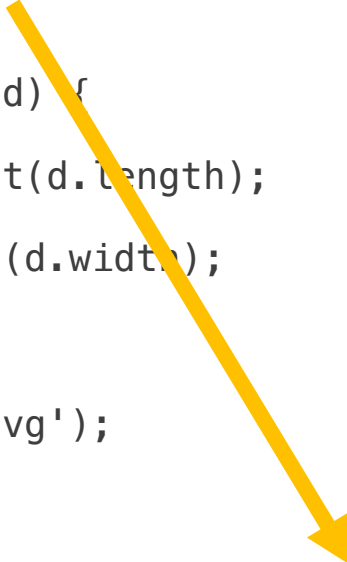
```
length width petalLength petalWidth species
5.1 3.5 1.4 0.2 setosa
4.9 3.0 1.4 0.2 setosa
4.7 3.2 1.3 0.2 setosa
4.6 3.1 1.5 0.2 setosa
5.0 3.6 1.4 0.2 setosa
5.4 3.9 1.7 0.4 setosa
```

산점도 만들기 (3)

```
d3.tsv('../data/iris.tsv',  
function(error, data) {  
  data.forEach(function(d) {  
    d.length = parseFloat(d.length);  
    d.width = parseFloat(d.width);  
  });  
  var svg = d3.select('svg');  
  
  ...  
  
});
```

```
<svg>  
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



length	width	petalLength	petalWidth	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

산점도 만들기 (4)

```

d3.tsv('../data/iris.tsv',
function(error, data) {
    data.forEach(function(d) {
        d.length = parseFloat(d.length);
        d.width = parseFloat(d.width);
    });
    var svg = d3.select('svg');
    ...

});

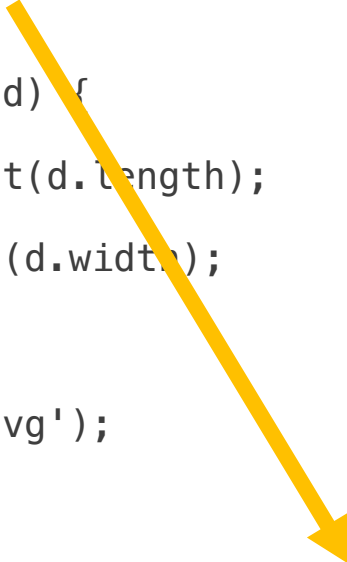
```

```

<svg>
</svg>

```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



length	width	petalLength	petalWidth	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

브라우저의 보안 정책으로 인해 로컬 스크립트
파일에서 외부 서버 파일 불러오기가 불가능

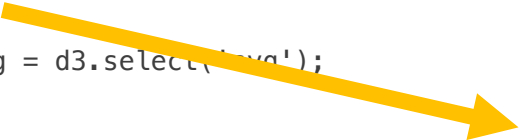
데이터 로딩 코드는 미리 작성되어 있으니 아래부터
작성하시면 됩니다.

산점도 만들기 (5)

```
data.forEach(function(d) {  
  d.length = parseFloat(d.length);  
  d.width = parseFloat(d.width);  
});  
var svg = d3.select('svg');
```

```
<svg>  
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



```
[  
  {"length": 5.1, "width": 3.5, "species": "setosa"},  
  ...  
]
```

산점도 만들기 (6)

```
data.forEach(function(d) {  
  d.length = parseFloat(d.length);  
  d.width = parseFloat(d.width);  
});  
var svg = d3.select('svg');
```

svg

```
.selectAll('circle')  
.data(data)  
.enter()  
  .append('circle')
```

```
<svg>  
  <circle></circle>  
  <circle></circle>  
  <circle></circle>  
  <circle></circle>  
  ...  
</svg>
```

산점도 만들기 (7)

```
svg
  .selectAll('circle')
  .data(data)
  .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return d.width * 100; })
    .attr('cy', function(d) { return d.length * 100; })
```

```
<svg>
  <circle r="3.5" cx="350" cy="510"></circle>
  ...
</svg>
```

```
[
  { "length": 5.1, "width": 3.5, "species": "setosa" },
  ...
]
```



산점도 만들기 (8)

1. 점의 위치를 상수 (100)을 곱하는 것이 아니라 조금 더 일반적으로 구할 수 있지 않을까?
2. X, Y 축을 보여주면 더 좋지 않을까?
3. 데이터에는 3가지 꽃의 종류가 있는데 종류에 따라 점의 색깔을 다르게 할 수 있지 않을까?



산점도 만들기 (9)

- Scale?
 - 데이터의 관측 값과 화면 상의 픽셀 값을 연결해주는 함수

Length (mm)

5.2
3.8
3.6

정의역 (domain)



Position (pixel)

260
190
180

치역 (range)

산점도 만들기 (10)

```
var width = 500, height = 500;
var x = d3.scale.linear()
    .domain([
        d3.min(data, function(d){return d.width;}),
        d3.max(data, function(d){return d.width;})
    ])
    .range([0, width]);

svg
    .selectAll('circle')
    .data(data)
    .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return x(d.width); })
    .attr('cy', function(d) { return d.length * 100; })
    .exit();
```

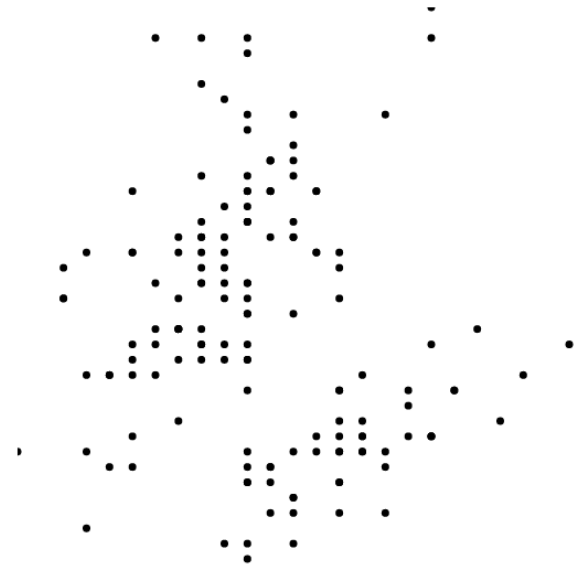


산점도 만들기 (11)

```
var width = 500, height = 500;
var x = d3.scale.linear()....;
var y = d3.scale.linear()
    .domain([
        d3.min(data, function(d){return d.length;}),
        d3.max(data, function(d){return d.length;})
    ])
    .range([height, 0]);
```

```
svg
    .selectAll('circle')
    .data(data)
    .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return x(d.width); })
    .attr('cy', function(d) { return y(d.length); })
    });
```

* 왼쪽 아래를 원점으로 만들기 위해
range를 거꾸로 넣었음에 주의



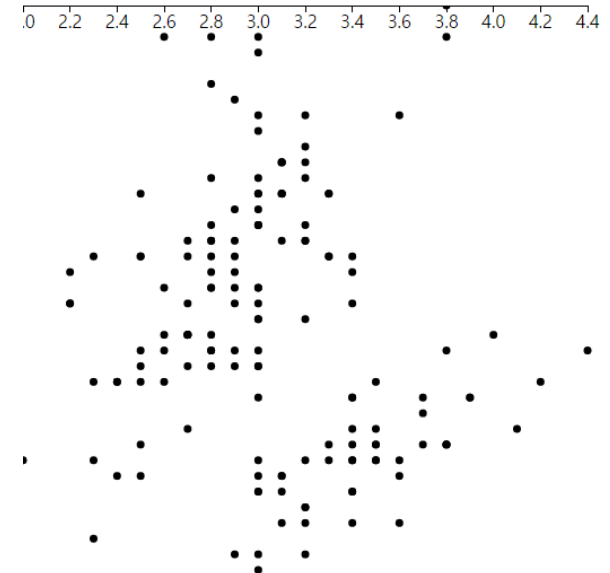
산점도 만들기 (12)

```
// .. 앞서 만든 코드에 추가
```

```
var x = d3.scale.linear()...;
```

```
var xAxis = d3.svg.axis()  
    .scale(x)  
    .orient('bottom');
```

```
svg  
    .append('g')  
    .attr('class', 'x axis')  
    .attr('transform', 'translate(0,' + height + ')')  
    .call(xAxis)
```



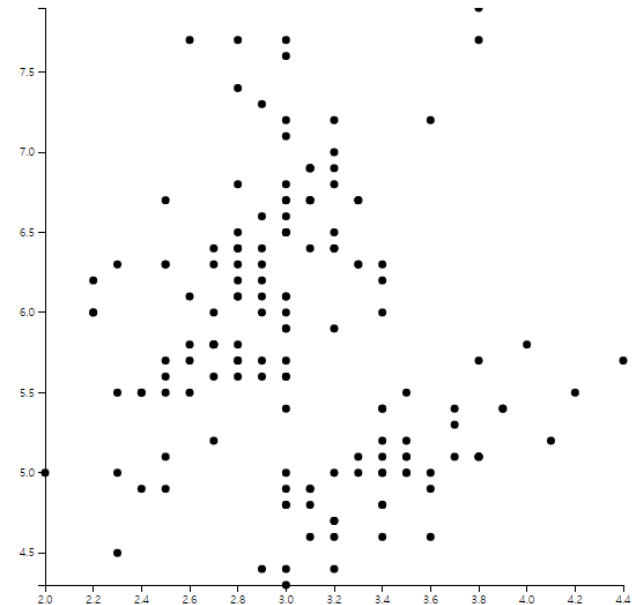
산점도 만들기 (13)

```
// .. 앞서 만든 코드에 추가
```

```
var yAxis = d3.svg.axis()  
    .scale(y)  
    .orient('left');
```

```
svg  
    .append('g')  
    .attr('class', 'y axis')  
    .attr('transform', 'translate(50, 0)')  
    .call(yAxis)
```

Y축을 표시할 공간을 확보하기 위해 x 스케일의 range
를 [0, width] 가 아닌 [50, width + 50] 으로 바꿔준다

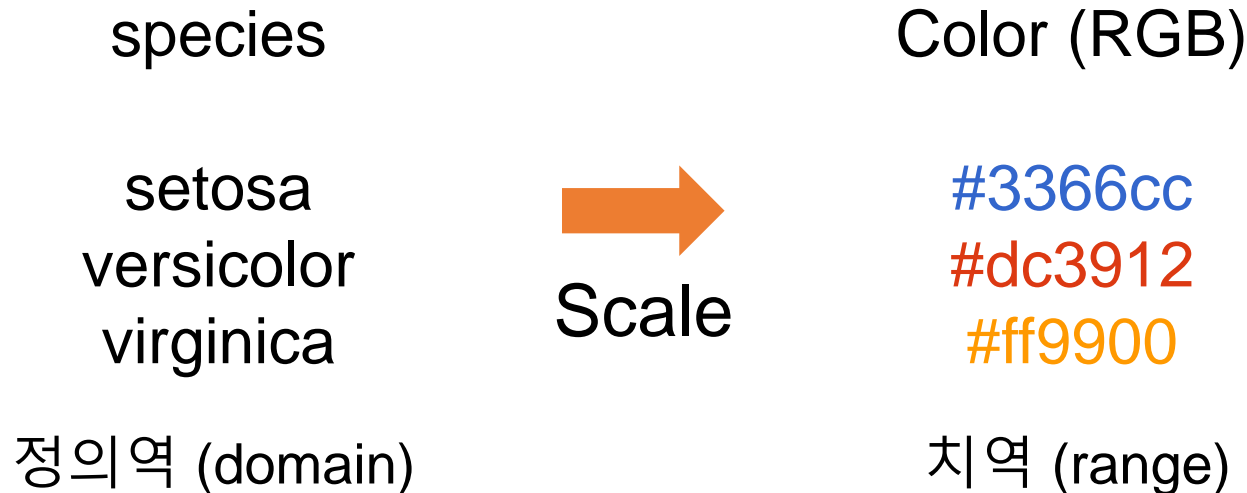


```
var x = d3.scale.linear()  
    ...  
    .range([50, width + 50]) // range([0, width])
```

산점도 만들기 (14)

• Color Scale?

- 데이터의 관측 값과 화면 상의 픽셀-값컬러코드를 연결해주는 함수



산점도 만들기 (15)

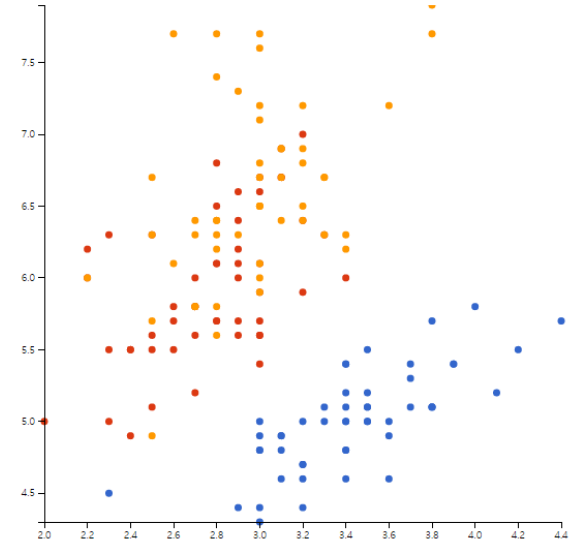
```

var width = 500, height = 500;
var x = d3.scale.linear()...;
var y = d3.scale.linear()...;

var color = d3.scale.ordinal()
    .domain(['setosa', 'versicolor', 'virginica'])
    .range(['#3366cc', '#dc3912', '#ff9900'])

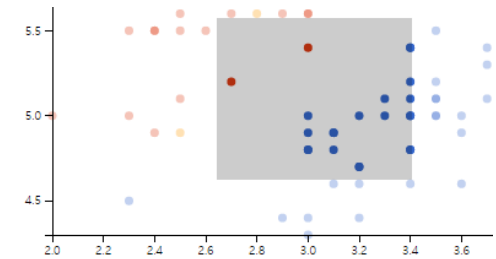
svg
    .selectAll('circle')
    .data(data)
    .enter()
        .append('circle')
        .attr('r', 3.5)
        .attr('cx', function(d) { return x(d.width); })
        .attr('cy', function(d) { return y(d.length); })
        .style('fill', function(d) { return color(d.species); })
    });

```



산점도 만들기 (16)

- 브러싱이란 분석자가 시각화의 관심있는 일부를 선택할 수 있도록 하여 보다 심화된 분석을 도와주는 인터랙션
- 산점도에서 관심있는 점들만 선택하고 선택된 점들의 평균 너비와 길이를 보이도록 해봅시다.



MEAN(width) = 3.20, MEAN(length) = 5.00



산점도 만들기 (17)

```
var brush = d3.svg.brush()
```

```
  .x(x)
```

```
  .y(y)
```

- 브러시 생성 후 x축 및 y축 스케일 지정
- 이를 통해 d3에서 스크린 좌표를 데이터 값으로 변환 가능

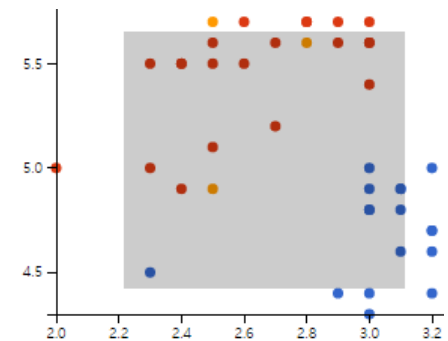
```
svg
```

```
  .append('g')
```

```
  .attr('class', 'brush')
```

```
  .call(brush)
```

- 새로운 g 엘리먼트를 추가하고 앞서 만든 brush 를 적용



MEAN(width) = , MEAN(length) =

산점도 만들기 (18)

```
var brush = d3.svg.brush()  
    .x(x)  
    .y(y)  
    .on('brush', update)  
    .on('brushend', update)
```

- 사용자가 드래그를 하거나 끝내는 이벤트를 감지하여 이때 선택된 점들의 평균 계산

```
function update() {  
    // ...  
}
```

산점도 만들기 (20)

```
function update() {  
    var extent = brush.extent();  
    var widthRange = [extent[0][0], extent[1][0]];  
    var lengthRange = [extent[0][1], extent[1][1]];  
    var widthSum = 0, lengthSum = 0, n = 0;  
  
    // ...  
}
```

- 브러싱 된 영역을 가져옴
- X축상의 선택된 꽃받침 너비의 범위
- Y축상의 선택된 꽃받침 길이의 범위
- 평균을 구하기 위해 변수 초기화

산점도 만들기 (21)

```
function update() {  
  // ...  
  svg  
    .selectAll('circle')  
    .style('opacity', 0.3)  
    .filter(function(d){  
      return widthRange[0] <= d.width && d.width <= widthRange[1] &&  
        lengthRange[0] <= d.length && d.length <= lengthRange[1];  
    })  
    .style('opacity', 1)  
    .each(function(d){  
      n++;  
      widthSum += d.width;  
      lengthSum += d.length;  
    })  
  // ...  
}
```

- 전체 점 선택
- 우선 모든 점을 반투명하게 함
- 현재 브러시에 속하는 점들만 선택
- 선택된 점들은 불투명하게 함
- 선택한 점들의 개수 및 길이와 너비의 합을 구함

산점도 만들기 (22)

```
function update() {  
  // ...  
  
  if(n > 0){  
    d3.select('#mean-width').text(d3.format('.2f')(widthSum / n));  
    d3.select('#mean-length').text(d3.format('.2f')(lengthSum / n));  
  }  
}
```

- 하나 이상의 점이 선택되었다면 너비와 길이의 평균을 소수 둘째 점 자리까지 표시

Network Diagram 만들기 (1)

- 노드 (node)
 - ex) 소셜 네트워크에서 유저
- 링크 (link)
 - ex) 두 사람의 친구관계
- Force-directed layout algorithm
 - 노드와 링크 사이에 작용하는 힘을 시뮬레이션하여 가장 안정적인 레이아웃을 찾아냄
 - 실행할 때 마다 결과가 달라짐



Network Diagram 만들기 (2)

1. 3_network 디렉토리 안에 있는 network_skeleton.js를 텍스트 에디터로 열기
 2. network_skeleton.html을 브라우저에서 열기
 3. js 파일을 수정한 후 저장하고 브라우저를 새로고침하여 결과를 확인
- 모든 html파일은 실습 내내 수정하실 필요가 없습니다.
 - 완성된 코드는 network.js에 있습니다.

Network Diagram 만들기 (3)

- 사용할 데이터: data/miserables.json
- 레 미제라블 등장 인물들의 관계 데이터
- nodes: 각 등장인물의 이름(name)과 그룹(group)
- links: 두 인물이 함께 등장하는 횟수(value)
- 등장인물들의 관계를 한눈에 볼 수 있는 네트워크 그래프를 그려 봅시다.

```
"nodes":[
  {"name": "Myriel", "group": 1},
  {"name": "Napoleon", "group": 1},
  {"name": "Mlle.Baptistine", "group": 1},
  {"name": "Mme.Magloire", "group": 1},
  {"name": "CountessdeLo", "group": 1},

  "links":[
    {"source": 1, "target": 0, "value": 1},
    {"source": 2, "target": 0, "value": 8},
    {"source": 3, "target": 0, "value": 10},
    {"source": 3, "target": 2, "value": 6},
    {"source": 4, "target": 0, "value": 1},
```

Network Diagram 만들기 (4)

```
var force = d3.layout.force()  
    .charge(-120)  
    .linkDistance(30)  
    .size([width, height]);
```

- force-directed layout 을 계산하는 인스턴스 생성
- charge: 노드들 사이에 작용하는 인력의 크기
 - 음의 값은 노드들끼리 서로 밀어냄
- linkDistance: 링크의 기본 길이
- size: 전체 시각화의 크기를 설정. 이 값을 이용해 그래프가 중앙으로 움직이도록 힘이 작용함

Network Diagram 만들기 (5)

```
// ...
```

```
d3.json('data/miserables.json',  
function(error, graph) {  
    force  
        .nodes(graph.nodes)  
        .links(graph.links)  
        .start();  
});
```

- 데이터를 불러옴
- 그래프의 노드와 링크를 설정
- 시뮬레이션 시작

Network Diagram 만들기 (6)

```
// ...  
  
var link = svg.selectAll('line')  
    .data(graph.links)  
    .enter().append('line')  
    .style('stroke-width', 2);  
  
var node = svg.selectAll('circle')  
    .data(graph.nodes)  
    .enter()  
    .append('circle')  
    .attr('r', 5)  
    .style('fill', function(d) {  
        return color(d.group);  
    })  
    .call(force.drag);
```

- 링크의 경우 line으로 그림
- Line의 기본 두께는 2px로 설정
- 노드의 경우 circle로 그림
- 반지름을 5px로 설정
- 각 그룹별로 노드에 다른 색을 칠함
- 기본적으로 제공하는 드래그 기능으로 노드를 움직일수 있도록 설정

Network Diagram 만들기 (7)

```
// ...  
force.on('tick', function() {  
  link  
    .attr('x1', function(d) { return d.source.x; })  
    .attr('y1', function(d) { return d.source.y; })  
    .attr('x2', function(d) { return d.target.x; })  
    .attr('y2', function(d) { return d.target.y; });  
  
  node  
    .attr('cx', function(d) { return d.x; })  
    .attr('cy', function(d) { return d.y; });  
});
```

- 시뮬레이션이 업데이트 될 때마다 불리는 callback 지정
- Callback 안에서는 업데이트된 좌표로 노드와 링크의 위치를 재설정



D3.js와 Apache Spark 연동하기

개요

- D3.js: 웹 브라우저에서 실행되는 시각화 라이브러리
 - 기술 스택의 최 상단에 위치
- Apache Spark: 서버측 분산 in-memory 컴퓨팅 플랫폼
 - 기술 스택의 하단에 위치
- 두 사이를 매개할 레이어들이 필요하다.
 - 웹 서버, job server, ...
 - 그러나 아직까지 모두를 통합한 플랫폼은 없음

D3.js

Web server

Job server

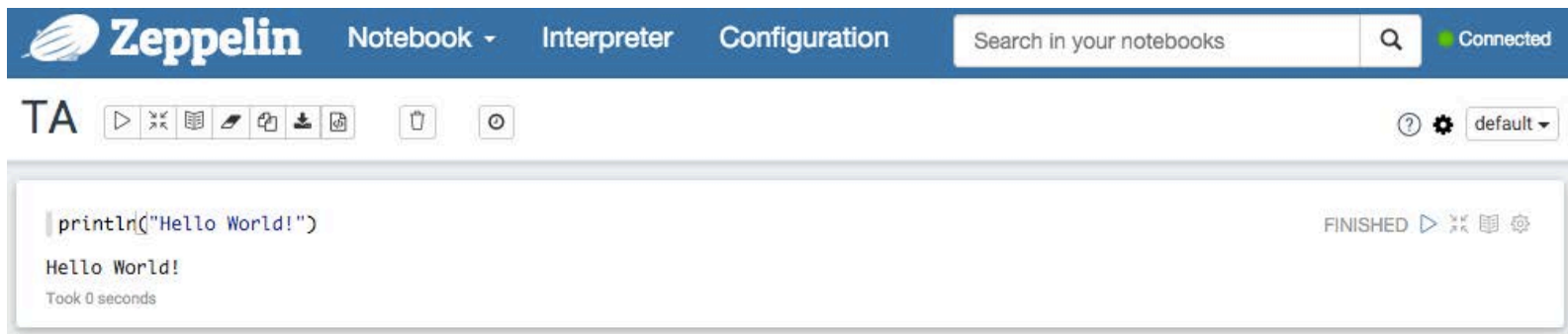
Apache Spark

개요

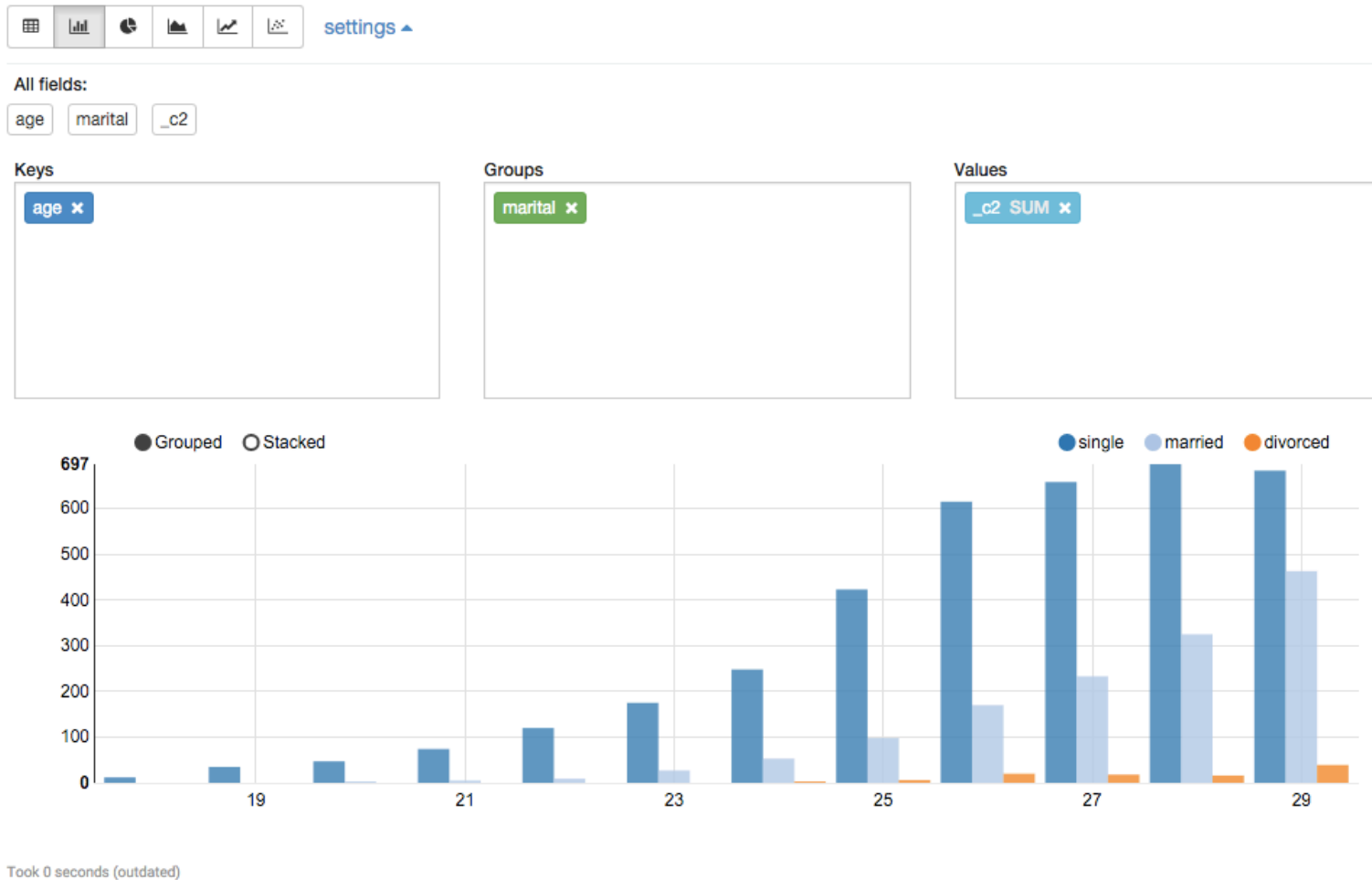
- 현재로서 가능한 방법
 1. 상용 플랫폼 (Tableau 등) 이용
 2. Apache Zeppelin 활용하기
 3. Apache Spark로 전처리하고 결과를 시각화하기
 4. 통합 시각적 분석 플랫폼 구축

Apache Zeppelin 소개

- Apache Spark와 같은 클러스터 컴퓨팅 엔진들과 연결해 사용할 수 있는 정보 시각화 프로그램
- 2014년 12월에 Apache 재단의 incubating이 시작되어, 현재 오픈 소스로 관리 중
 - <https://github.com/apache/incubator-zeppelin>



Apache Zeppelin 소개



Apache Zeppelin 소개

- 장점

- 가장 간단하게 Apache Spark와 연동하여 코드를 작성하고 그 결과를 시각화해 볼 수 있음
- 기본적인 시각화 및 인터랙션 지원
 - 바 차트, 산점도, 라인 차트 등
 - 피벗 변환, 툴팁 등

- 단점

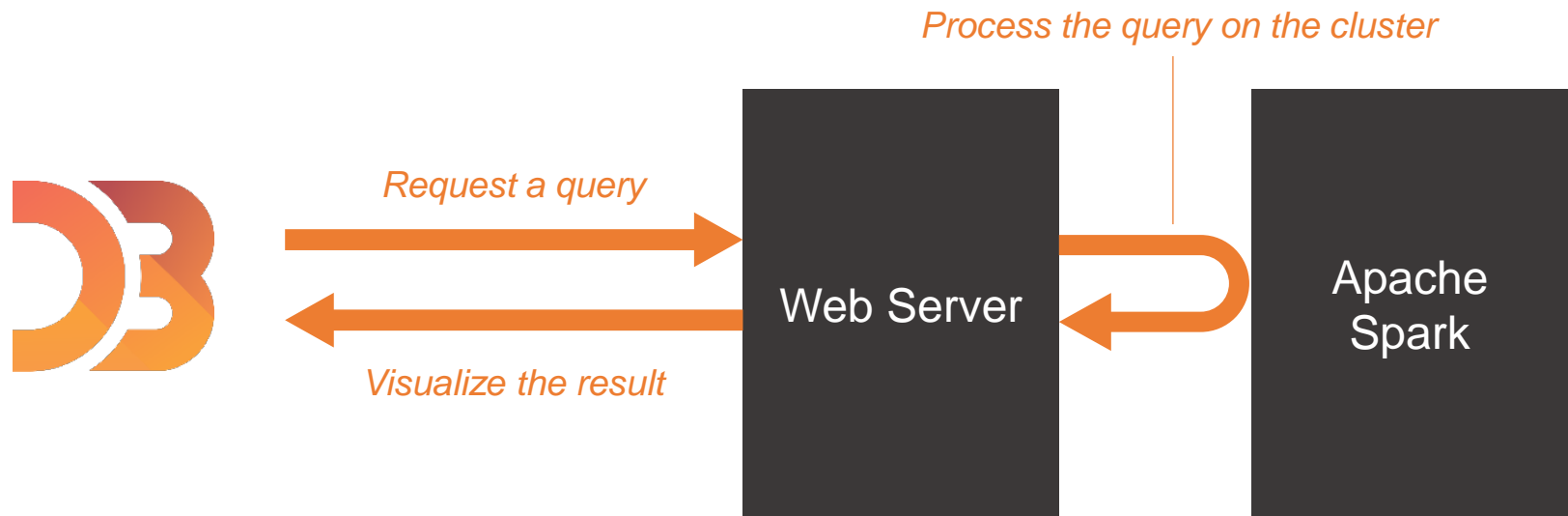
- 아직 개발중이기 때문에 설치 및 유지 보수가 까다로움
- 새로운 시각화를 추가하기 힘들

Apache Spark로 전처리 후 결과를 시각화

- Apache Spark에서 계산한 것을 CSV 혹은 JSON 형태로 저장하고 이를 분석 시나리오에 알맞은 형태로 시각화
- 장점
 - 가장 간단하고 직관적인 방법
 - Apache Spark 외에 추가 플랫폼 관리 필요 없음
- 단점
 - 데이터를 탐색하면서 다양한 쿼리를 수행해 보려면 Spark 드라이버 프로그램을 수정해야 함

시각적 분석 플랫폼 구축

- D3.js를 이용하여 Apache Spark로 분석한 결과를 시각화하는 플랫폼 구축
 - 웹 인터페이스를 통한 인터랙션으로 데이터 탐색
 - Apache Spark와 d3.js의 강점을 둘다 살릴 수 있음
 - 웹 서버, 잡 서버 등의 추가적인 레이어 필요
 - <https://github.com/spark-jobserver/spark-jobserver>



참고자료

- 다양한 예제 및 소스코드
 - <https://github.com/mbostock/d3/wiki/Gallery>
- API 명세
 - <https://github.com/mbostock/d3/wiki/API-Reference>
 - <https://github.com/zziuni/d3/wiki>
- 튜토리얼 및 읽을거리
 - <http://alignedleft.com/tutorials/d3>
 - <https://www.dashingd3js.com/table-of-contents>
 - <http://www.jaeminjo.com/>

수고하셨습니다.