

Apache Zeppelin + D3.js

서울대학교 컴퓨터공학부
인간-컴퓨터 상호작용 연구실
김원재 조재민

실습 자료

- 본 발표 자료 및 실습 자료들은 아래 Github 저장소에서 받으실 수 있습니다.

<https://github.com/dandelin/SNUBDA>

- 실습 시간 이후의 질문 사항은 위 Github 저장소에 이슈로 등록해주시길 바랍니다.



Apache Zeppelin

Apache Zeppelin이란?

- 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 Flink, Apache Tajo, Cassandra 등의 엔진과도 연결이 가능하지만 아직 완벽하지 않으므로 본 실습에서는 Apache Spark와 연결해서 사용합니다.

Apache Zeppelin 설치

- 2015년 2월 현재 Apache Zeppelin은 Binary 설치파일을 제공하지 않으며, Mac OSX과 Linux를 지원합니다.
- 따라서 Zeppelin을 설치하려면 Github의 README를 참조하여 소스 코드를 직접 빌드해야 합니다.
- 하지만 소스 코드를 빌드하는 과정이 오래 걸리고 예기치 못한 에러가 발생할 수 있으니 본 실습에서는 Zeppelin이 미리 설치되어 있는 AWS 서버를 사용합니다.

Apache Zeppelin의 장점

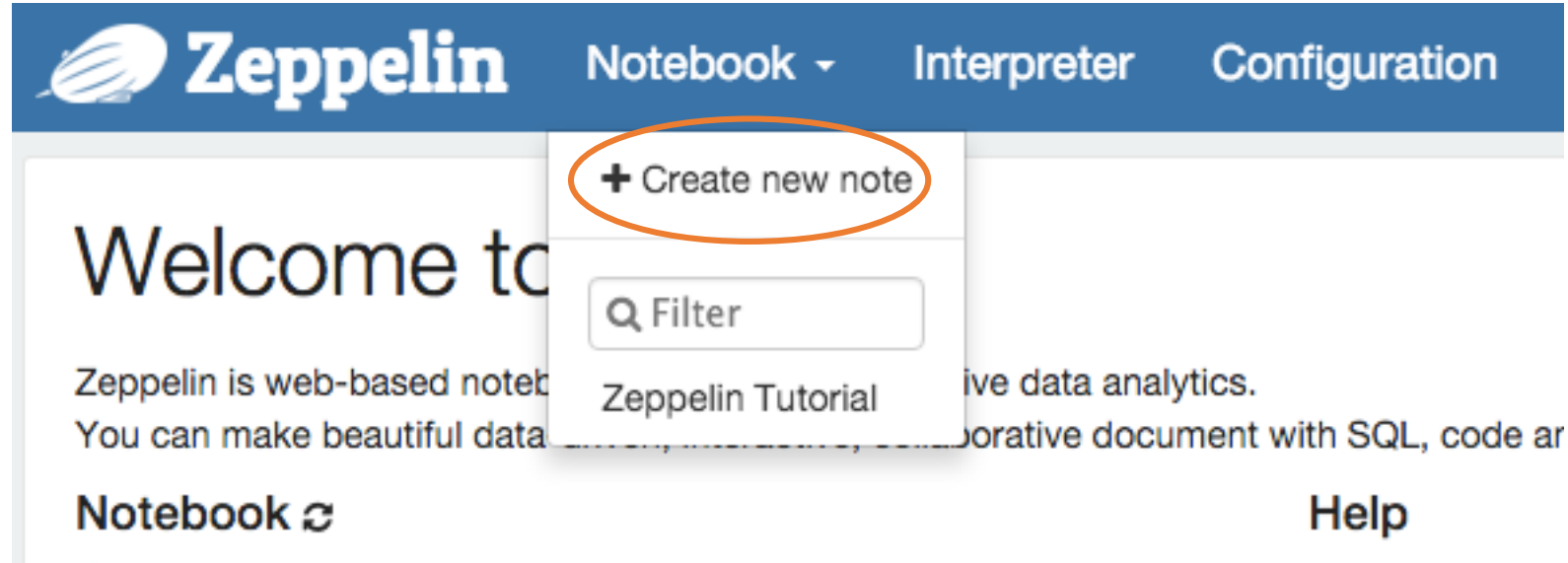
- 기존의 빅 데이터 분석 및 시각화의 workflow는 아래와 같음
 1. 데이터 저장 → Hadoop Distributed File System (HDFS)
 2. 데이터 정제, 처리 → Hadoop MapReduce, HIVE 등
 3. 시각화를 위한 데이터 추리기 → MySQL, MongoDB 등
 4. 시각화 → Tableau, Spotfire 등
 5. 분석 → R, mahout 등
- Zeppelin은 위 workflow중 데이터 저장을 제외한 모든 부분을 한번에 수행할 수 있음

데이터 분석에 앞서 (1)

- 본 실습에서는 외부 클러스터와 연동하지 않고, 로컬에 있는 스파크를 사용함. 외부 클러스터와 연동을 원하는 경우 환경 변수 파일의 MASTER 변수를 수정하여 사용
 - <http://zeppelin-project.org/docs/install/install.html> 페이지의 configure 참조
- 4.6MB의 크기를 가진 은행 계좌 데이터를 사용
 - [Moro et al., 2011] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.

데이터 분석에 앞서 (2)

- 실습 시간에 공지된 주소로 접속
- Notebook 드롭다운 메뉴를 통해 새 노트 생성
 - 주의 : 다른 사람의 노트에 접속하지 말아주세요.



데이터 분석에 앞서 (3)

- 다른 사람 노트와 헷갈리지 않게 본인 이름으로 작성해주세요

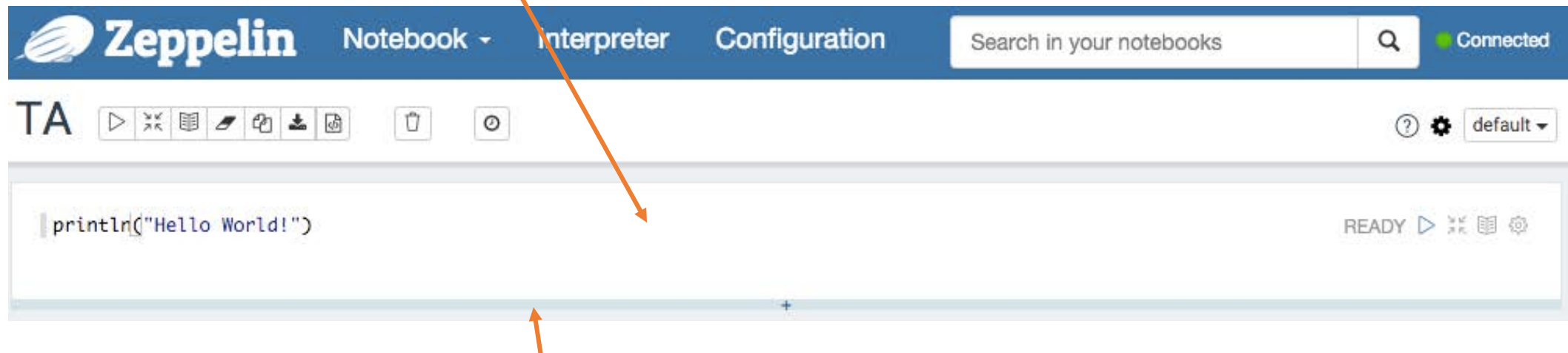
Create new note ×

Note Name

Create Note

노트 구성 요소 (1)

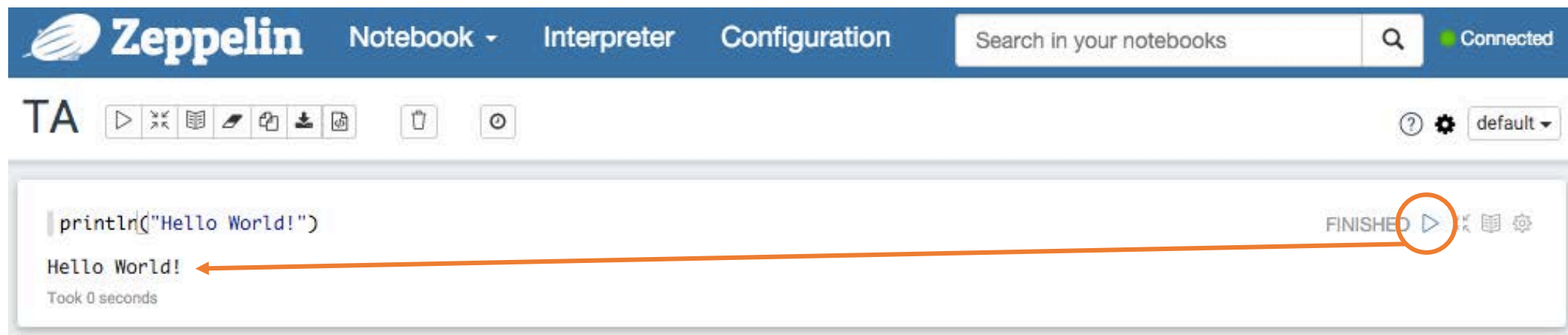
- 노트는 문단(Paragraph)의 연속



- 문단의 위 아래 모서리에 마우스를 올리면 문단을 추가할 수 있다.

노트 구성 요소 (2)

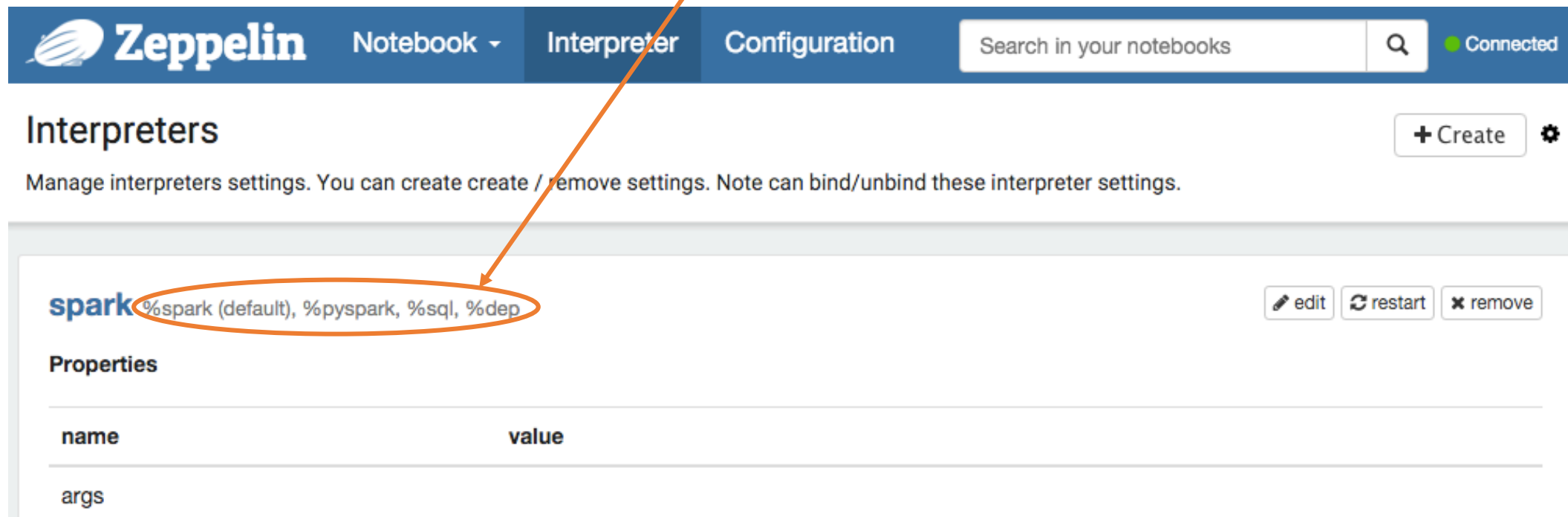
- 문단은 실행 가능한 코드의 블록



- 문단 시작에 아무런 indicator를 쓰지 않으면 그 문단을 실행할 시에 쓰여있는 코드를 Scala with SparkContext 코드로 해석하여 실행함(REPL)

노트 구성 요소 (3)

- 문단 시작 시 %sql과 같은 indicator를 쓰면 해당 코드로 해석하여 실행함, indicator의 목록은 상단 네비게이션 바에서 Interpreter를 선택하여 확인할 수 있음



Zeppelin Notebook ▾ Interpreter Configuration Search in your notebooks 🔍 Connected

Interpreters

+ Create ⚙️

Manage interpreters settings. You can create / remove settings. Note can bind/unbind these interpreter settings.

spark
%spark (default), %pyspark, %sql, %dep

edit
restart
remove

Properties

name	value
args	

데이터 구조

- bank-full.csv의 구조는 오른쪽과 같다.
- 첫번째 row에는 ;으로 구분된 각 column의 이름이 쓰여있다.
- 두번째 row부터는 ;으로 구분된 해당 row의 각 column값이 쓰여있다.

```
bank-full.csv
1 "age";"job";"marital";"education";"default";"balance";"housing";
  "loan";"contact";"day";"month";"duration";"campaign";"pdays";"pr
  evious";"poutcome";"y"
2 58;"management";"married";"tertiary";"no";2143;"yes";"no";"unkno
  wn";5;"may";261;1;-1;0;"unknown";"no"
3 44;"technician";"single";"secondary";"no";29;"yes";"no";"unknown
  ";5;"may";151;1;-1;0;"unknown";"no"
4 33;"entrepreneur";"married";"secondary";"no";2;"yes";"yes";"unkn
  own";5;"may";76;1;-1;0;"unknown";"no"
5 47;"blue-collar";"married";"unknown";"no";1506;"yes";"no";"unkno
  wn";5;"may";92;1;-1;0;"unknown";"no"
6 33;"unknown";"single";"unknown";"no";1;"no";"no";"unknown";5;"ma
  y";198;1;-1;0;"unknown";"no"
7 35;"management";"married";"tertiary";"no";231;"yes";"no";"unknow
  n";5;"may";139;1;-1;0;"unknown";"no"
8 28;"management";"single";"tertiary";"no";447;"yes";"yes";"unknow
  n";5;"may";217;1;-1;0;"unknown";"no"
9 42;"entrepreneur";"divorced";"tertiary";"yes";2;"yes";"no";"unkn
  own";5;"may";380;1;-1;0;"unknown";"no"
10 58;"retired";"married";"primary";"no";121;"yes";"no";"unknown";5
    ;"may";50;1;-1;0;"unknown";"no"
```

데이터 정제 (1)

- Scala Paragraph에서 sc (SparkContext)의 parallelize 메서드나 다른 방식을 통해 사용할 데이터를 RDD로 만든다. 본 실습에서는 sc.textFile 메서드로 csv로부터 RDD를 만듦

```
val bankText = sc.textFile("/home/ubuntu/bank/bank-full.csv")
```

```
bankText: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[51] at textFile at <console>:26
```

```
Took 0 seconds
```

데이터 정제 (2)

- Bank Case Class를 생성한다.
 - Case Class란 스칼라가 지원하는 Constructor의 parameter가 노출된 Class이다
 - 17개 column중에서 5개만 선택

```
case class Bank(age:Integer, job:String, marital : String, education : String, balance : Integer)
```

```
defined class Bank
```

Took 0 seconds

데이터 정제 (3)

- bankText RDD의 각 행을 Bank Class로 변환한다.
 - 각 string을 ;으로 split하여 쪼개진 첫번째 인자가 “age” 인 것만 filter한 뒤(스키마 열 제외) 각 쪼개진 배열을 사용하여 모든 따옴표를 제거한 뒤 Bank Constructor의 인자로 적절히 넣는다. 0번째, 5번째 인자인 age, balance는 int로 변환해준다.

```
val bank = bankText.map(s=>s.split(";")).filter(s=>s(0)!="age").map(
  s=>Bank(s(0).toInt,
    s(1).replaceAll("\"", ""),
    s(2).replaceAll("\"", ""),
    s(3).replaceAll("\"", ""),
    s(5).replaceAll("\"", "").toInt
  )
)
```

```
bank: org.apache.spark.rdd.RDD[Bank] = MapPartitionsRDD[39] at map at <console>:30
```

```
Took 1 seconds
```


데이터 정제 (4)

- bank RDD의 toDF 메서드를 사용하여 RDD를 Data Frame으로 변환하고 Data Frame의 registerTempTable 메서드를 사용하여 bank라는 이름의 in-memory table을 생성한다.

```
bank.toDF().registerTempTable("bank")
```

Took 0 seconds

데이터 탐색 (1)

- SQL Paragraph에서 등록한 bank table에 age < 30인 age를 가진 row가 각 (age, marital)별로 테이블에 얼마나 있는지 물어보는 SQL query를 작성하고 실행함

```
%sql select age, marital, count(1) from bank where age < 30 group by age, marital order by age
```

FINISHED ▶ ⌵ 📖 ⚙️

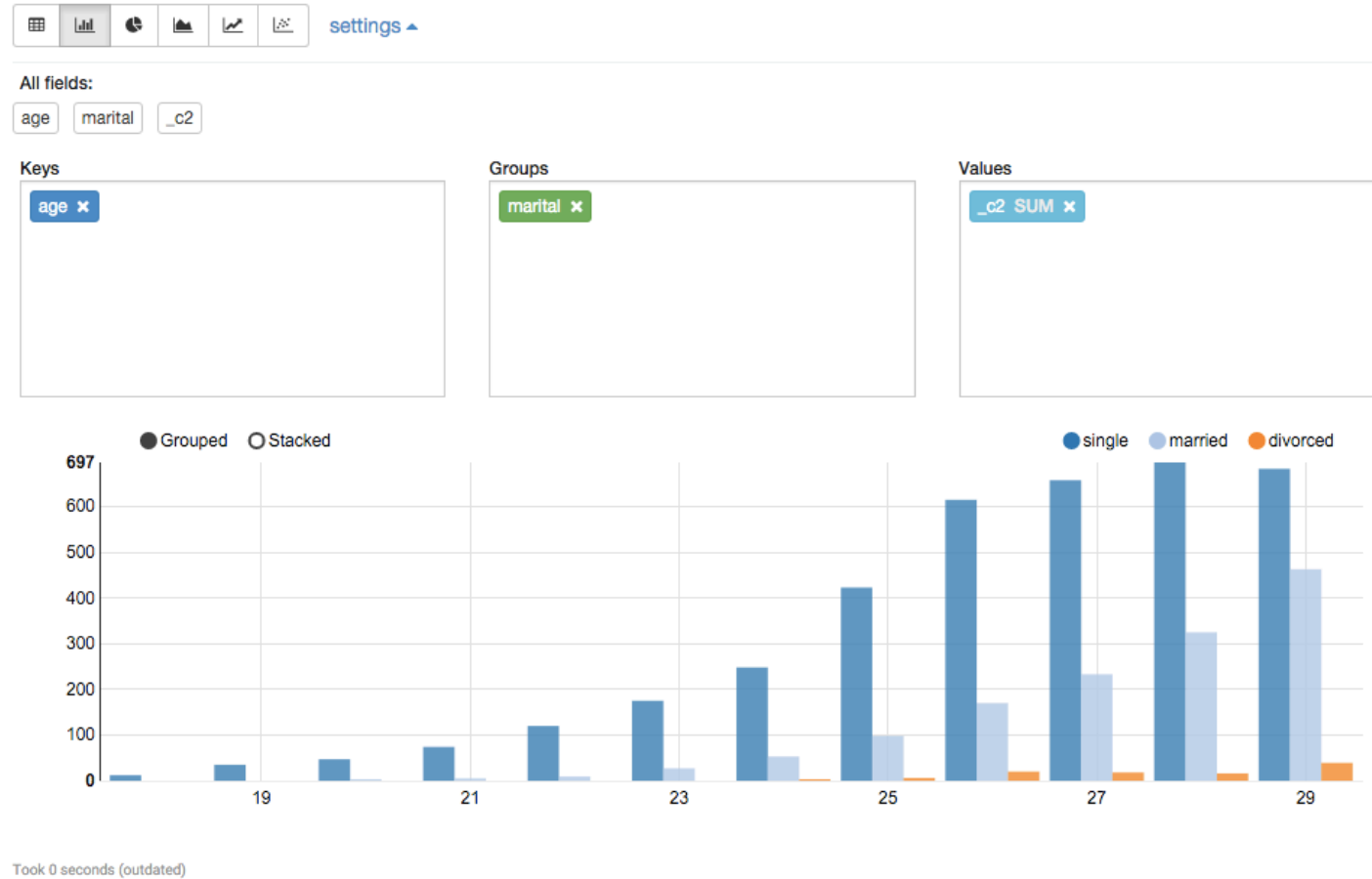


age	marital	_c2
18	single	12
19	single	35
20	single	47
20	married	3
21	married	5
21	single	74

데이터 탐색 (2)

- 가져온 row들은 Zeppelin이 자동으로 Table, {Bar, Pie, Area, Line} Chart 그리고 Scatter Plot으로 시각화를 함
- 각 차트의 옵션(settings)에서 결과 컬럼을 차트의 어떤 요소에 바인딩할지 정할 수 있음
 - Key, Group, or Value
- 예시로 지금 가져온 데이터에서 나이대별로 결혼 상태에 대한 비율을 보고 싶다면 Bar Chart에서 Age를 key로, Marital을 group으로, count를 value로 지정해주면 됨

데이터 탐색 (3)



Apache Zeppelin의 한계 (1)

- 현재 개발 중인 상황이기 때문에 바이너리가 존재하지 않아 설치가 어렵고, 모든 OS에서 동작하지 않는다.
- 기존의 빅 데이터 분석 및 시각화의 Workflow의 각 단계를 능숙하게 다룰 수 있을 때 할 수 있는 작업들을 완전히 재현하는 것은 불가능하다.

Apache Zeppelin의 한계 (2)

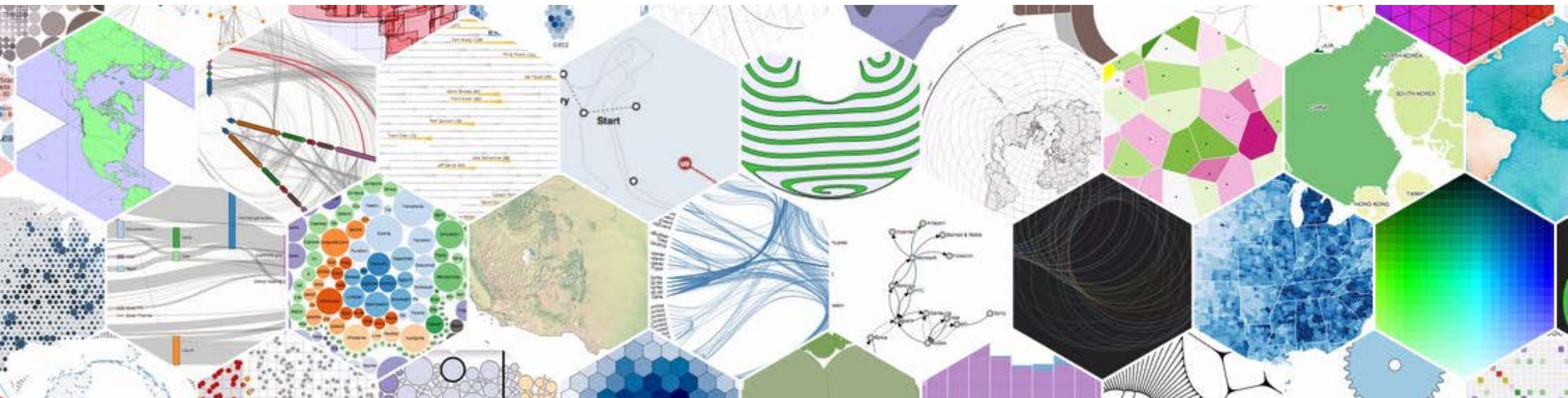
- 특히, 정보 시각화는 제공되는 시각화 중에서만 선택해야 한다는 한계를 지닌다.
 - 예를 들어 소셜 데이터에 대해서 네트워크 다이어그램이나 Co-occurrence Martix를 이용하는 등의 데이터에 특화된 분석이 불가능하다.
- 조금 더 일반적인 데이터 시각화 방법?

D3.js

D3.js 실습에서는..

1. D3.js 소개
2. D3.js의 핵심 아이디어 설명
3. 막대그래프 그려보기
4. 산점도 그려보기
5. Network Diagram 그려보기
6. (참고) 빈도수 매트릭스 그려보기

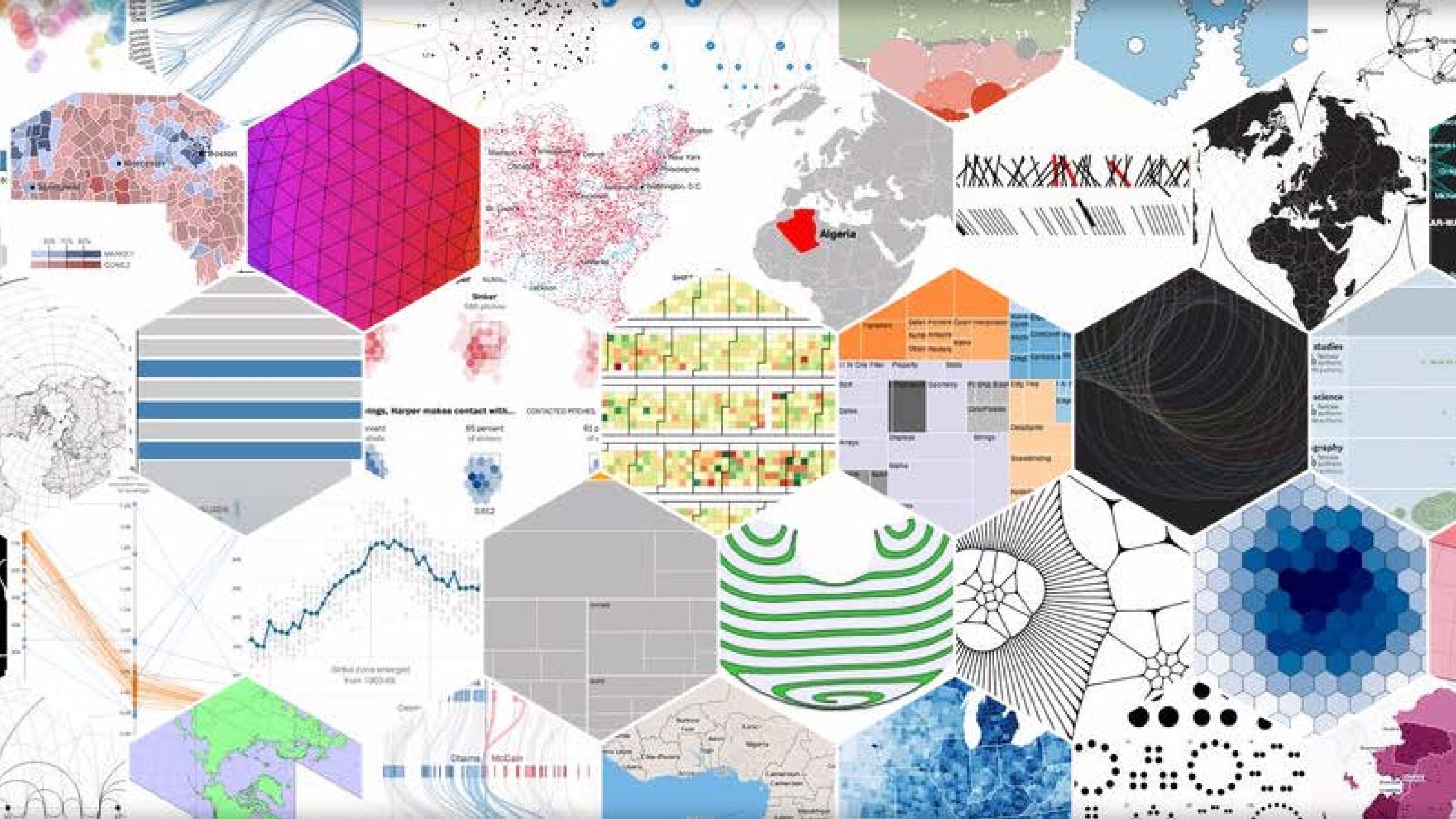
D3.js is JavaScript library for manipulating documents based on data – d3js.org



D3.js의 탄생 배경

- 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?



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
- 역동적인 커뮤니티
 - 현재 Github에서 사용자들이 5번째로 많이 Star한 라이브러리
- 웹 플랫폼: 별도의 설치가 필요 없고 쉽게 접근 가능함

Why D3.js?

- Highcharts, CharJS, Plotly, Pygal, ... 수많은 시각화 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

실습 전에 알고있으면 좋은 것들

- HTML, CSS
 - 웹 기반 시각화이므로 웹 문서를 구성하는 기본 요소들에 대한 사전 지식이 있으면 좋음
- Javascript
 - D3.js는 자바스크립트 라이브러리이므로 자바스크립트에 대한 선행 지식이 있으면 좋음

D3.js 사용 준비하기 (1)

- 실습자료 다운로드 받기

<https://github.com/dandelin/SNUBDA>

D3.js 사용 준비하기 (2)

- 준비물: 웹 브라우저와 텍스트 에디터
- 웹 브라우저
 - 본 실습에서는 Chrome을 기준으로 설명
 - D3.js에 크로스 브라우징 기능이 내장되어 있으므로 다른 브라우저도 가능
 - IE의 경우 로컬 스크립트 실행여부를 묻는 알람창이 뜨면 “차단된 콘텐츠 허용” 클릭

D3.js 사용 준비하기 (3)

- 텍스트 에디터
 - Sublime, Vim, Emacs, Visual Studio, ...
 - <http://www.sublimetext.com/2>
- 발표자료에서 코드 복사 시 따옴표가 유니코드 따옴표로 바뀌는 문제가 있으니 주의
 - 코드 복사는 완성된 파일을 참고해서 해주세요.

D3.js 사용 준비하기 (4)

- 실습 폴더에는 네 개의 예제가 독립된 폴더로 있음
 1. 막대그래프 (bar chart)
 2. 산점도 (scatterplot)
 3. 네트워크 그래프 (network diagram)
 4. 빈도수 행렬 (co-occurrence matrix)
- 한번 둘러 보시다.

D3.js의 핵심 아이디어 (1)

	Value
Sally	5
Tom	10
John	7.5

Table

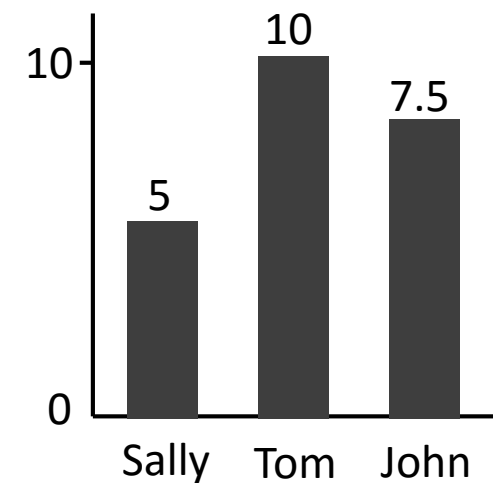


Chart
(Visualization)

D3.js의 핵심 아이디어 (2)

	Value
Sally	5
Tom	10
John	7.5

Table

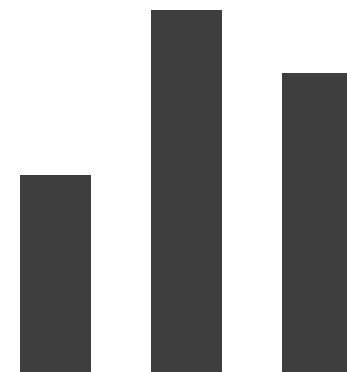


Chart
(Visualization)

D3.js의 핵심 아이디어 (3)

	Value
Sally	5
Tom	10
John	7.5

Table

```
[
  {"Name": "Sally",
   "Value": 5},
  {"Name": "Tom", "
   Value": 10},
  {"Name": "John",
   "Value": 7.5}
]
```

JSON
(csv, tsv, ...)

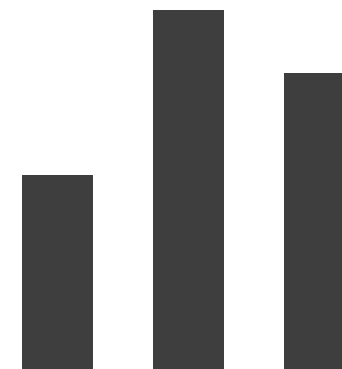


Chart
(Visualization)

D3.js의 핵심 아이디어 (4)

	Value
Sally	5
Tom	10
John	7.5

Table

```
[
  {"Name": "Sally",
   "Value": 5},
  {"Name": "Tom", "
   Value": 10},
  {"Name": "John",
   "Value": 7.5}
]
```

JSON

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

SVG
(Scalable
Vector
Graphics)

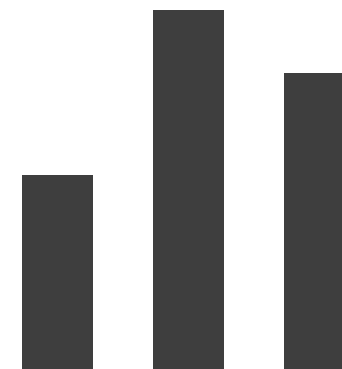


Chart
(Visualization)

D3.js의 핵심 아이디어 (5)

	Value
Sally	5
Tom	10
John	7.5

Table

```
[
  {"Name": "Sally",
   "Value": 5},
  {"Name": "Tom", "
   Value": 10},
  {"Name": "John",
   "Value": 7.5}
]
```

JSON

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

D3.js

SVG
(Scalable
Vector
Graphics)

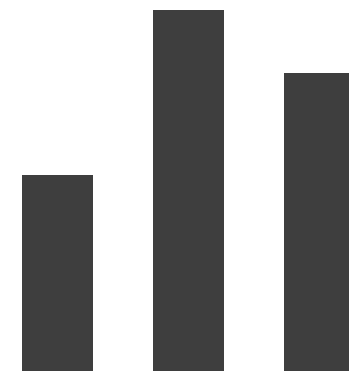


Chart
(Visualization)

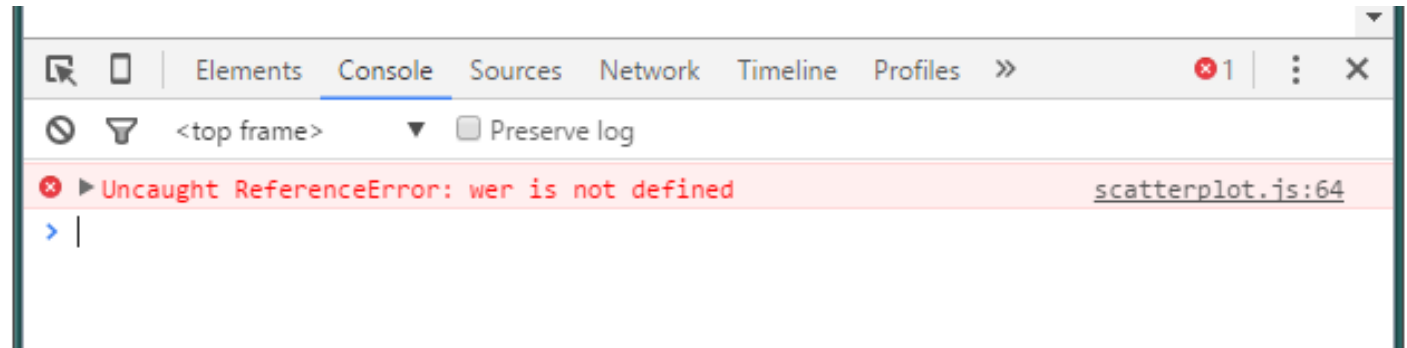
D3.js로 바 차트 그리기 (1)

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.

크롬 개발자 도구 사용

- Windows Chrome에서는 F12, Mac에서는 Command+Alt+I 로 개발자 도구 실행

- [Console] 탭으로 이동



- 코드에 에러가 생기면 에러 종류 및 위치를 알 수 있다.

D3.js로 바 차트 그리기 (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

D3.js로 바 차트 그리기 (3)

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

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

```
<svg>
```

```
</svg>
```

Javascript

HTML

D3.js로 바 차트 그리기 (4)

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

```
<svg>
```

```
</svg>
```

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

```
svg
```

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


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

Javascript

HTML

D3.js로 바 차트 그리기 (5)

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



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

Javascript

HTML

D3.js로 바 차트 그리기 (6)

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

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

```
svg
```

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

```
  .data(data)
```

```
  .enter()
```

```
<svg>
```

```
</svg>
```

Javascript

HTML

D3.js로 바 차트 그리기 (7)

var data = ...		<svg>
	{“name”:“Sally”, “value”:5}	•
var svg = d3.select(‘svg’)	{“name”:“Tom”, “value”:10}	•
	{“name”:“John”, “value”:7.5}	•
svg		</svg>
.selectAll(‘rect’)		
.data(data)		
.enter()		

Javascript

HTML

D3.js로 바 차트 그리기 (8)

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

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

```
svg
```

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

```
.data(data)
```

```
.enter()
```

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

```
{ "name": "Sally", "value": 5 }
```

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

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

```
<svg>
```

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

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

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

```
</svg>
```

자바스크립트로 HTML 요소를 생성
(코드 입력 X)

Javascript

HTML

D3.js로 바 차트 그리기 (9)

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

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

```
svg
```

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

```
.data(data)
```

```
.enter()
```

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

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

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

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

```
{ "name": "Sally", "value": 5 }
```

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

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

```
<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>
```



Javascript

HTML

D3.js로 바 차트 그리기 (10)

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

svg                            {"name": "Tom", "value": 10}
  .selectAll('rect')
  .data(data)                  {"name": "John", "value": 7.5}
  .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>
```



D3.js로 바 차트 그리기 (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 + ')';
    });
```

```
{"name": "Sally", "value": 5}
```

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

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

```
<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>
```

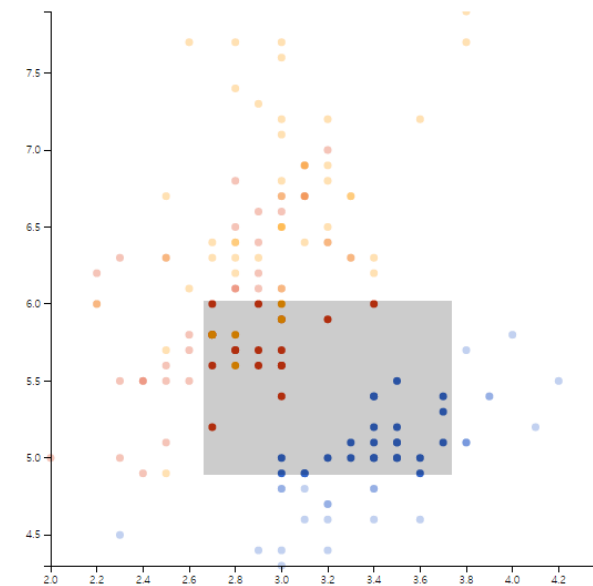
```
</svg>
```



D3.js로 산점도 그리기 (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

D3.js로 산점도 그리기 (2)

- 사용할 데이터: data/iris.tsv

```
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
```

- 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.

D3.js로 산점도 그리기 (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
```

```
});
```

Javascript

HTML

D3.js로 산점도 그리기 (4)

```
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

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

데이터 로딩 코드는 미리 작성되어 있으니 아래부터 작성하시면 됩니다.

Javascript

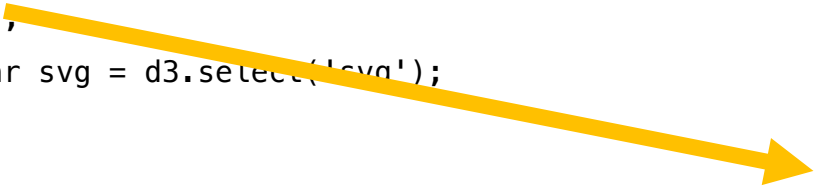
HTML

D3.js로 산점도 그리기 (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"},
  ...
]
```

Javascript

HTML

D3.js로 산점도 그리기 (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')
```

Javascript

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

HTML

D3.js로 산점도 그리기 (7)

```
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')
    .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 개선하기

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?



Scale 사용하기 (1)

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

Length (mm)

5.2

3.8

3.6

정의역 (domain)



Scale

Position (pixel)

260

190

180

치역 (range)

Scale 사용하기 (2)

```
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; })
    });
```

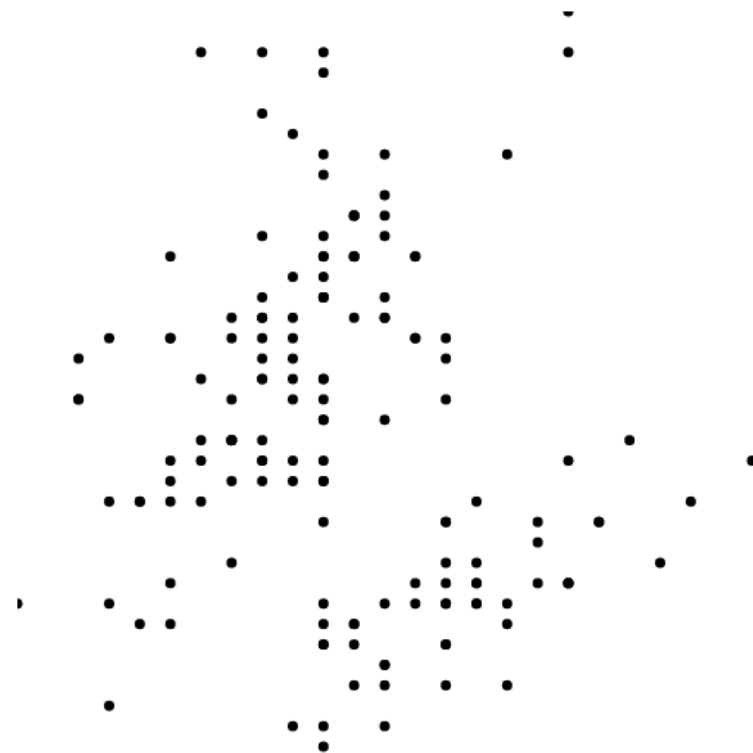


Scale 사용하기 (3)

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

* 왼쪽 아래를 원점으로 만들기 위해
range를 거꾸로 넣었음에 주의

```
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); })
  .attr('fill', function(d) { return color(d.length); })
  .attr('stroke', function(d) { return color(d.length); })
  .attr('stroke-width', 1);
```



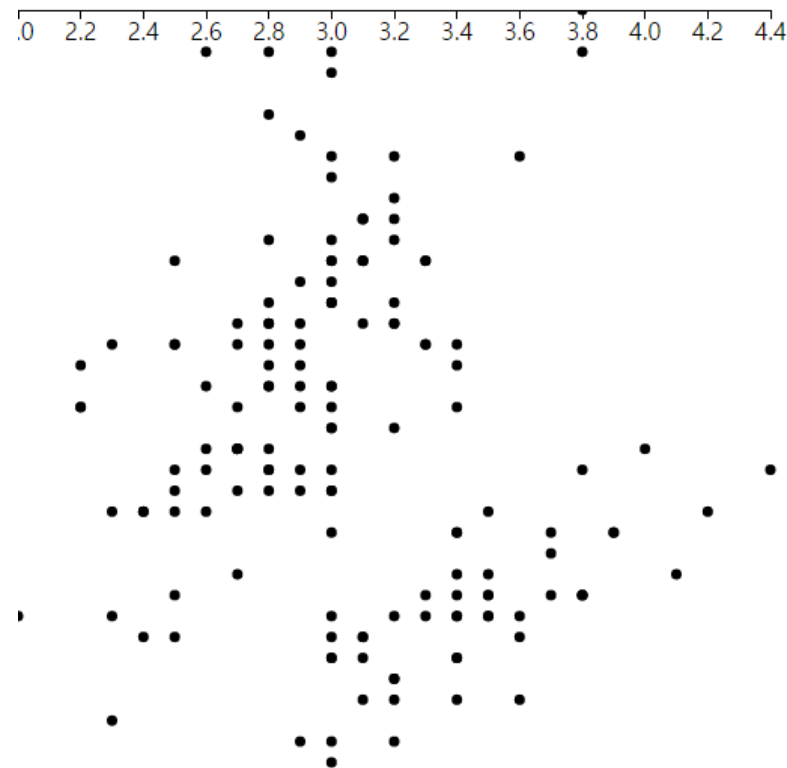
X, Y 축 넣기 (1)

// .. 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')  
  .call(xAxis)
```



X, Y 축 넣기 (2)

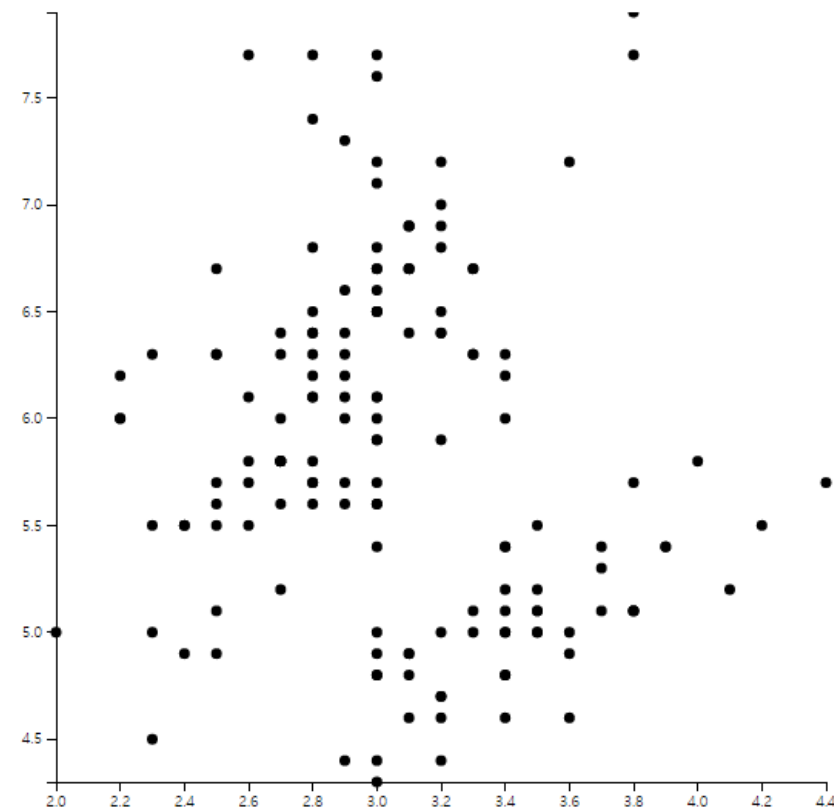
```
// .. 앞서 만든 코드에 추가
```

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

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

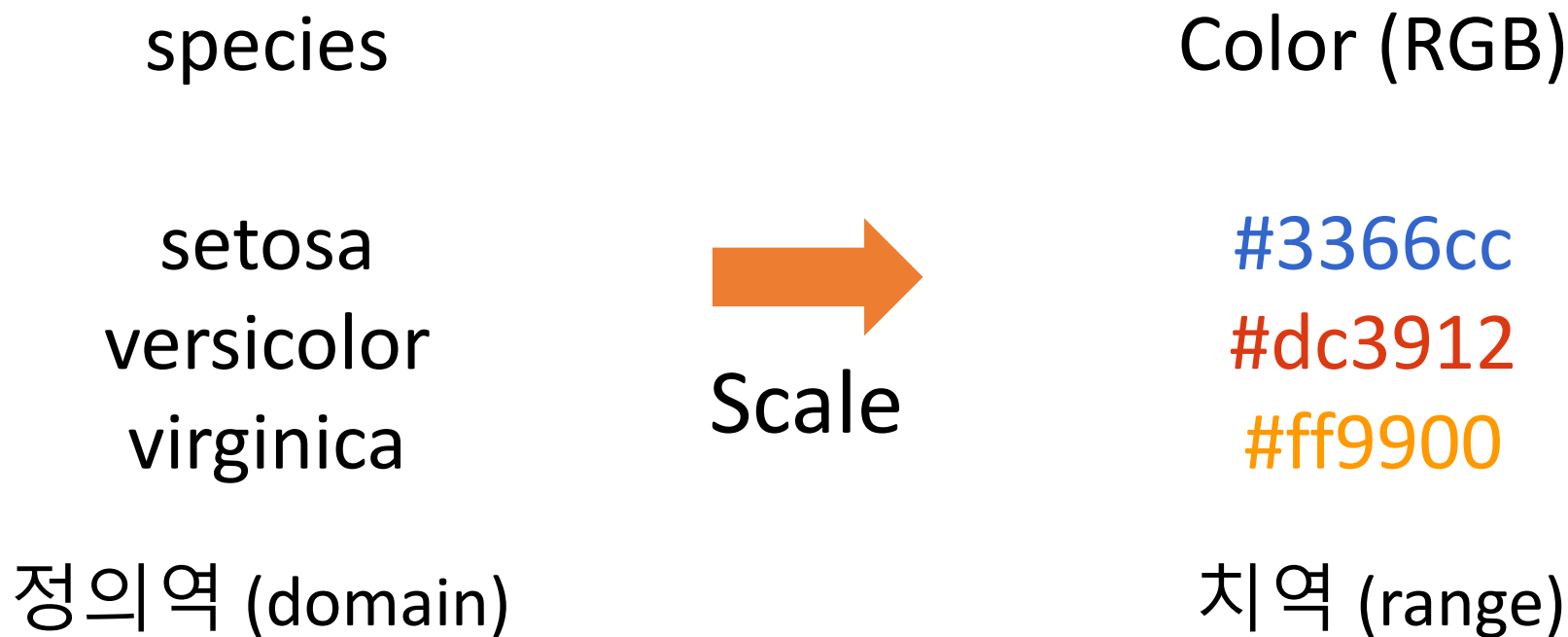
Y축을 표시할 공간을 확보하기 위해 x 스케일의 range를
[0, width] 가 아닌 [50, width + 50] 으로 바꿔준다

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



Color Scale 사용하기

- Color Scale?
 - 데이터의 관측 값과 화면 상의 픽셀-값컬러코드를 연결해주는 함수

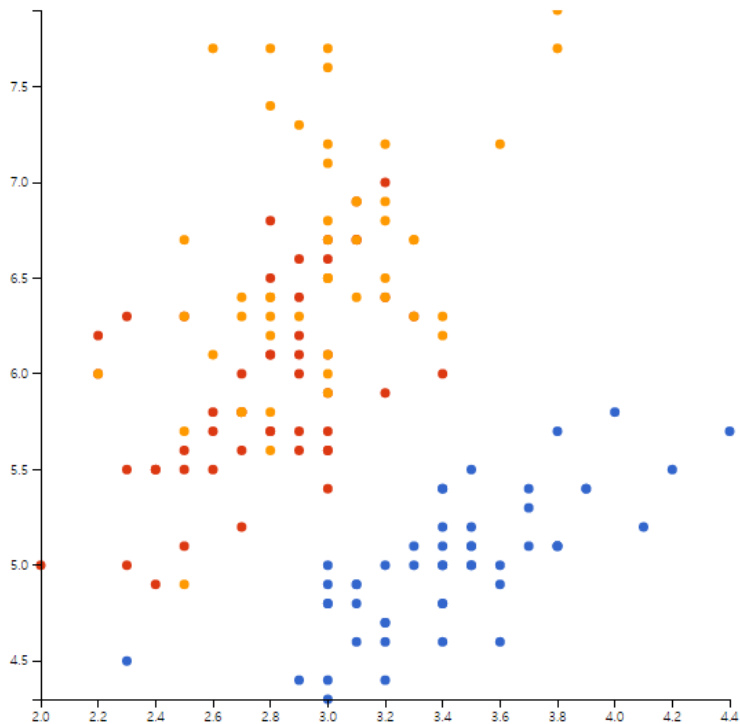


Scale 사용하기

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

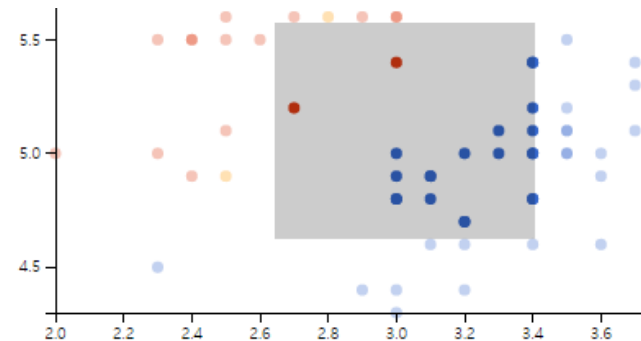
var color = d3.scale.ordinal()
    .domain(['setosa', 'versicolor', 'virginica'])
    .range(['#3366cc', '#dc3912', '#ff9900'])

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); })
    });
```



Brushing 인터랙션 추가하기

- 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



Brush 사용하기 (1)

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

```
.x(x)
```

```
.y(y)
```

```
svg
```

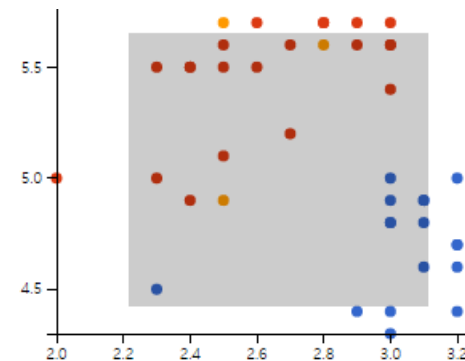
```
.append('g')
```

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

```
.call(brush)
```

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

- 새로운 g 엘리먼트를 추가하고 앞서 만든 brush를 적용



MEAN(width) = , MEAN(length) =

Brush 사용하기 (2)

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

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

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

Brush 사용하기 (3)

```
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

Brush 사용하기 (4)

```
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

Brush 사용하기 (5)

```
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

- 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



D3.js로 Network Diagram 그리기 (1)

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.

D3.js로 Network Diagram 그리기 (2)

- 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},
```

D3.js로 Network Diagram 그리기 (3)

```
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

D3.js로 Network Diagram 그리기 (4)

```
// ...
```

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

- 데이터를 불러옴
- 그래프의 노드와 링크를 설정
- 시뮬레이션 시작

D3.js로 Network Diagram 그리기 (5)

```
// ...
var link = svg.selectAll('line')
    .data(graph.links)
    .enter().append('line')
    .style('stroke-width', 2);

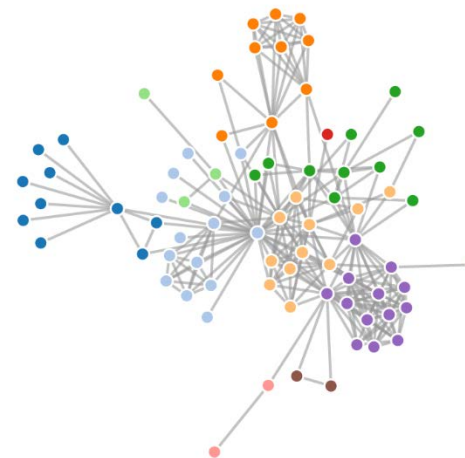
var node = svg.selectAll('circle')
    .data(graph.nodes)
    .enter()
    .append('circle')
    .attr('r', 5)
    .style('fill', function(d) {
        return color(d.group);
    })
    .call(force.drag);
```

- 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

D3.js로 Network Diagram 그리기 (6)

```
// ...  
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



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.